

## 4 Understanding ABI Files

---

The Application Binary Interface (ABI) is a JSON-description of the methods for transforming user actions between JSON and binary representations. ABI also describes methods for converting database state to JSON-format and vice versa. After describing the contract through ABI, developers and users can easily interact with it via JSON.

To create an ABI file, use the `eosio-cpp` utility located in `cyberway.cdt`. Sometimes, for debugging purposes or for analysis, it is necessary to modify manually an already created ABI file. Therefore, the developers need to familiarize themselves with its structure, as well as with a description of its individual components. The structure of an ABI file is the following:

```
1  {
2    "version": "cyberway::abi/1.1",
3    "structs": [],
4    "types": [],
5    "actions": [],
6    "events": [],
7    "tables": [],
8    "variants": [],
9    "abi_extensions": []
10 }
```

**structs** — a description of the structures used to process the actions of the user. The structures can also be used to describe the fields of objects that are stored in tables or the fields (parameters) of functions in actions.

**types** — a description of user types used as parameters in any public action of the user or structure. It can also be used across public actions or structures. You may also describe your own types by yourself.

**actions** — a description of the public actions of the user that combines all the functions described in the header file of the contract.

**events** — a description of events, each of which represents a reaction (response) of a contract to a specific action of the user.

**tables** — a description of the tables used to process the actions of the user. The tables are objects of the type `multi_index` or `eosio::singleton`, which store data in the database of the blockchain node. Inside the table are `structs`. Creating, receiving, modifying or deleting objects is performed in `actions`.

**abi\_extensions** — a description of additional ABI features (reserved, not used).

---

## Structs

An example of a structure description in an ABI file:

```
1 {
2     "name": "issue", // The name
3     "base": "",      // Inheritance, parent struct
4     "fields": []      // Array of field objects describing the struct's fields
5 }
```

An example of a structure field description in an ABI file:

```
1 {
2     "name": "",      // The field's name
3     "type": ""       // The field's type
4 }
```

An example of a structure of a `create` operation for creating tokens in an ABI file:

```
1 {
2     "name": "create",
3     "base": "",
4
5     "fields": [
6         {
7             "name": "issuer", // Token creator name
8             "type": "name"
9         },
10        {
```

```
11     "name": "maximum_supply", // Maximum number of tokens
12     "type": "asset"
13 }
14 ]
15 }
```

---

## Types

Пример описания типа в ABI-файле:


```
1 {
2     "new_type_name": "name",
3     "type": "name"
4 }
```

---

## Actions

An example of a type description in an ABI file:

```
1 {
2     "name": "transfer",      // The name of the action as defined in the contract
3     "type": "transfer"      // The name of the implicit struct as described in the contract
4 }
```



---

## Events

An example of the description of events for a publishing contract in an ABI file:

```
1 "events": [
2     {"name": "poolstate", "type": "pool_event"},
3     {"name": "poolerase", "type": "uint64"},
4 ]
```

```
4      {"name": "postcreate", "type": "create_event"},
5      {"name": "poststate", "type": "post_event"},
6      {"name": "postclose", "type": "post_close"},
7      {"name": "votestate", "type": "vote_event"}
8  ],
```

---

## Tables

An example of the description of a table in an ABI file:

```
1  {
2      "name": "emitparams",
3      "type": "emit_state",
4      "indexes" : [{
5          "name": "primary",
6          "unique": true,
7          "orders": [{"field": "id", "order": "asc"}]
8      }]
9  }
```

---

## Vectors' description

At first, you have to add a type using `[]` characters to describe a vector in an ABI-file. The ABI file tables must be accurately described. For example, in the case of cleos when adding a table to a contract with a distorted ABI description data may be lost if there is no error message popping out.

You have to update the ABI file straight after a single change is made in the contract structure, tables, actions or events, as well as after introduction of a new type. In most cases if the ABI file update fails, an error message may not appeared. Such situation is difficult to analyze the work of the contract.