# Session keys with meta-transactions

Meta-transactions (outside execution) allows external contracts to execute transactions from outside an account contract, thereby opening up new use-cases such as sponsored transactions, deploying an account on behalf of a user, transaction scheduling (limit orders) etc.

Get outside execution call

To retrieve the data needed to perform anexecute_from_outside:

- sessionRequest
- ,accountSessionSignature
- and the account address are needed
- importArgentSessionService
- andSessionDappService
- from@argent/x-sessions
- .
-
    - The first is required to instantiateSessionDappService
-
    - SessionDappService
-
    - expose the methodgetOutsideExecutionCall

```

Copy consttransferCallData=erc20Contract.populate("transfer",{ recipient:address, amount:parseInputAmountToUint256(amount), });

constbeService=newArgentSessionService( dappKey.publicKey, accountSessionSignature, ARGENT_SESSION_SERVICE_BASE_URL// Do not share publicly );

constsessionDappService=newSessionDappService(beService,awaitprovider.getChainId(),dappKey);

const{contractAddress,entrypoint,calldata}=awaitsessionDappService.getOutsideExecutionCall( sessionRequest, stark.formatSignature(accountSessionSignature), false,// keep false, will be a feat for caching [transferCallData], address, chainId, shortString.encodeShortString("ANY_CALLER"),// Optional: default value ANY_CALLER execute_after, // Optional: timestamp in seconds - this is the lower value in the range. Default value: 5 mins before Date.now() execute_before, // Optional: timestamp in seconds - this is the upper value in the range. Default value: 20 mins after Date.now() nonce: BigNumberish,// Optional: nonce, default value is a random nonce );

```

The result can be executed by another account or at backend level

Get outside execution TypedData

To retrieve the TypedData for an outside execution along with the compiled signature (session + guardian).

- sessionRequest
- ,accountSessionSignature
- and the account address are needed
- importArgentSessionService
- andSessionDappService
- from@argent/x-sessions
- .
-
    - The first is required to instantiateSessionDappService
-
    - SessionDappService
-
    - expose the methodgetOutsideExecutionTypedData
-
    -
        - it takes the same parameters asgetOutsideExecutionCall

---

- *
-

```

Copy consttransferCallData=erc20Contract.populate("transfer",{ recipient:address, amount:parseInputAmountToUint256(amount), });

constbeService=newArgentSessionService( dappKey.publicKey, accountSessionSignature, ARGENT_SESSION_SERVICE_BASE_URL );

const{signature,outsideExecutionTypedData}=awaitsessionDappService.getOutsideExecutionTypedData( sessionRequest, stark.formatSignature(accountSessionSignature), false, [transferCallData], address );

```

Session backend service

If needed,@argent/x-sessions provides thesessionBackenService to directly call the argent backend for signing a session or signing a session for en execution from outside

sign session

```

Copy public asyncsignTxAndSession( calls: Call[], transactionsDetail: InvocationsSignerDetails, sessionTypedData: TypedData, sessionSignature: bigint[], cacheAuthorisation: boolean, ):Promise<{ publicKey:string r:bigint s:bigint }

```

sign session for an execution from outside (EFO)

```

Copy public asyncsignSessionEFO( sessionTokenToSign: MySession, accountAddress: string, currentTypedData: TypedData, sessionSignature: bigint[], cacheAuthorisation: boolean, chainId: StarknetChainId, ):Promise<{ publicKey:string r:bigint s:bigint }

```

Previous Implement session keys Next Demo App

Last updated3 months ago