**TUTORIAL BÁSICO DE WALT-ID CLI**

[1. INTRODUÇÃO 1](#_Toc172135355)

# INTRODUÇÃO

É necessário ter a versão 15 do java sdk instalada. Escolha um diretório apropriado e rode os comandos:

***git clone https://github.com/walt-id/waltid-identity.git***

***cd waltid-identity***

***./gradlew clean build***

***cd waltid-applications/waltid-cli***

***alias waltid="./waltid-cli.sh"***

***waltid --help***

Rodando o comando “***waltid --help***” você observará as funcionalidades:

Sobre a geração de chaves, rodando o comando “waltid key generate --help” você encontra os seguintes algoritmos de geração de chaves disponíveis: Ed25519; secp256k1; RSA; secp256r1. O algoritmo padrão é o secp256r1, para definir outro basta usar o parâmetro “--keyType" ou “-t”, como no exemplo:

***waltid key generate --keyType=secp256k1***

Retornando o resultado:

*Generating key of type secp256k1...*

*Key thumbprint is: CUoJxRZmNfQ6nQ0RNFrCG3S7gSIGVjutvPWanYYM25k*

*Generated Key (JWK):*

*{*

*"kty": "EC",*

*"d": "BLxMHejfIaQZagJqStH0xC70POT6VJoxNY5jc6szYeA",*

*"crv": "secp256k1",*

*"kid": "CUoJxRZmNfQ6nQ0RNFrCG3S7gSIGVjutvPWanYYM25k",*

*"x": "-U2mAiYZvT66aK9bqA8CJxETpynfacKDKUemfEFGghA",*

*"y": "Inc5TablJvIt8xviBIYbKPa9o6v2rH1Av4rJONAt6as"*

*}*

*Done. Key saved at file "…/waltid-identity/waltid-applications/waltid-cli/CUoJxRZmNfQ6nQ0RNFrCG3S7gSIGVjutvPWanYYM25k.json".*

Agora vamos criar uma DID. Com o comando “***waltid did create --help***” verificamos que não há parâmetros obrigatórios, então pode-se criar uma DID apenas rodando “***waltid did create***”. Um importante parâmetro é o “*--method*” ou “*-m*”, que pode ter os valores: KEY, JWK, WEB, CHEQD, IOTA. Também é possível utilizar o parâmetro “*--key*” ou “*-k*” para definir uma chave criptográfica específica, caso este parâmetro não seja utilizado será criada uma nova chave especialmente para a nova DID. Utilizaremos a chave já criada neste exemplo:

***waltid did create --key=CUoJxRZmNfQ6nQ0RNFrCG3S7gSIGVjutvPWanYYM25k.json***

O retorno do prompt foi:

DID Subject key to be used:

{

"kty": "EC",

"d": "BLxMHejfIaQZagJqStH0xC70POT6VJoxNY5jc6szYeA",

"crv": "secp256k1",

"kid": "CUoJxRZmNfQ6nQ0RNFrCG3S7gSIGVjutvPWanYYM25k",

"x": "-U2mAiYZvT66aK9bqA8CJxETpynfacKDKUemfEFGghA",

"y": "Inc5TablJvIt8xviBIYbKPa9o6v2rH1Av4rJONAt6as"

}

DID Document:

{

"content": {

"@context": [

"https://www.w3.org/ns/did/v1",

"https://w3id.org/security/suites/jws-2020/v1"

],

"id": "did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91",

"verificationMethod": [

{

"id": "did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91#zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91",

"type": "JsonWebKey2020",

"controller": “did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91",

"publicKeyJwk": {

"kty": "EC",

"crv": "secp256k1",

"kid": CUoJxRZmNfQ6nQ0RNFrCG3S7gSIGVjutvPWanYYM25k",

"x": "-U2mAiYZvT66aK9bqA8CJxETpynfacKDKUemfEFGghA",

"y": "Inc5TablJvIt8xviBIYbKPa9o6v2rH1Av4rJONAt6as"

}

}

],

"assertionMethod": [

"did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91#zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91"

],

"authentication": [

"did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91#zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91"

],

"capabilityInvocation": [

"did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91#zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91"

],

"capabilityDelegation": [ "did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91#zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91"

],

"keyAgreement": [

"did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91#zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91"

]

}

}

DID created:

did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91

O próximo passo é assinar uma VC. O WaltID CLI não faz a criação de credenciais verificáveis, então será necessário pegar uma pronta e salvar em um arquivo. Templates de VCs podem ser encontradas no link [https://credentials.walt.id/credentials](https://credentials.walt.id/credentials/vaccinationcertificate), O modelo “vaccinationcertificate” foi escolhido e salvo no arquivo “myVC.json”.

Em uma VC há o envolvimento de duas DIDs: uma tem o papel de emissor (issuer) da credencial e a outra de titular (holder) da credencial. Rodando o comando “***waltid vc sign --help***” verificamos que não é necessário definir a DID do issuer, pois ela será gerada com base no método did:key presente no arquivo da credencial.

É obrigatório definir a DID do holder, uma chave criptográfica e o nome do arquivo em que a credencial está salva.

Neste exemplo reutilizaremos o que já foi obtido nos comandos anteriores:

“***waltid vc sign --Key=./CUoJxRZmNfQ6nQ0RNFrCG3S7gSIGVjutvPWanYYM25k.json --subject=did:key:zQ3shwRHKKoPfAqJFwFMiJWQgCKGxDkDj4ePxHDonDUdF7m91 ./myVC.json***”

Resultado:

Done. Signed VC saved at …/waltid-identity/waltid-applications/waltid-cli/./myVC.signed.json

Assim já é possível realizar a verificação com o comando:

“***waltid vc verify ./myVC.signed.json***”

Resultado:

*Verification Result*

*signature: Success!*

O comando “***waltid vc verify***” dispõe da opção “*-p*” ou “*--policy*” para que múltiplas políticas sejam verificadas. As políticas disponíveis são: signature, expired, not-before, revoked\_status\_list, schema, allowed-issuer e webhook. É possível criar os seus próprios esquemas e customizar ainda mais a verificação. Exemplo de comando utilizando políticas:

“***waltid vc verify -p expired -p not-before -p signature ./myVC.signed.json***”

*Verification Result*

*expired: Fail! VC expired since 2022-08-31T00:00:00Z*

*not-before: Success!*

*signature: Success!*

Mais informações: <https://github.com/walt-id/waltid-identity/tree/main/waltid-applications/waltid-cli>