The principle of MFA

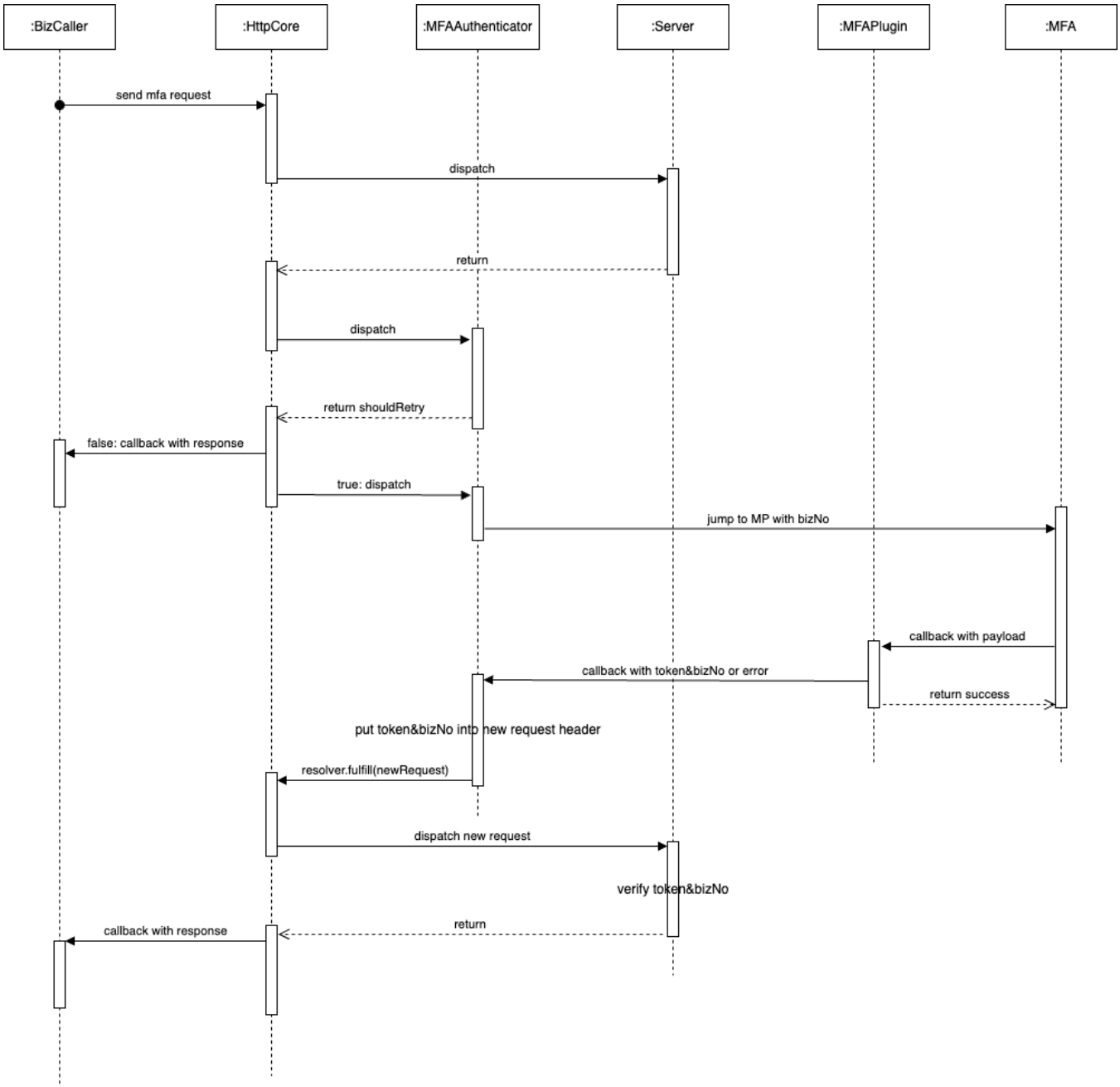
1 、 Description

Do you know how to call MFA? the fact is that we just need to call API, which is okay. as shown in the following code. let us think think how to achieve it ?

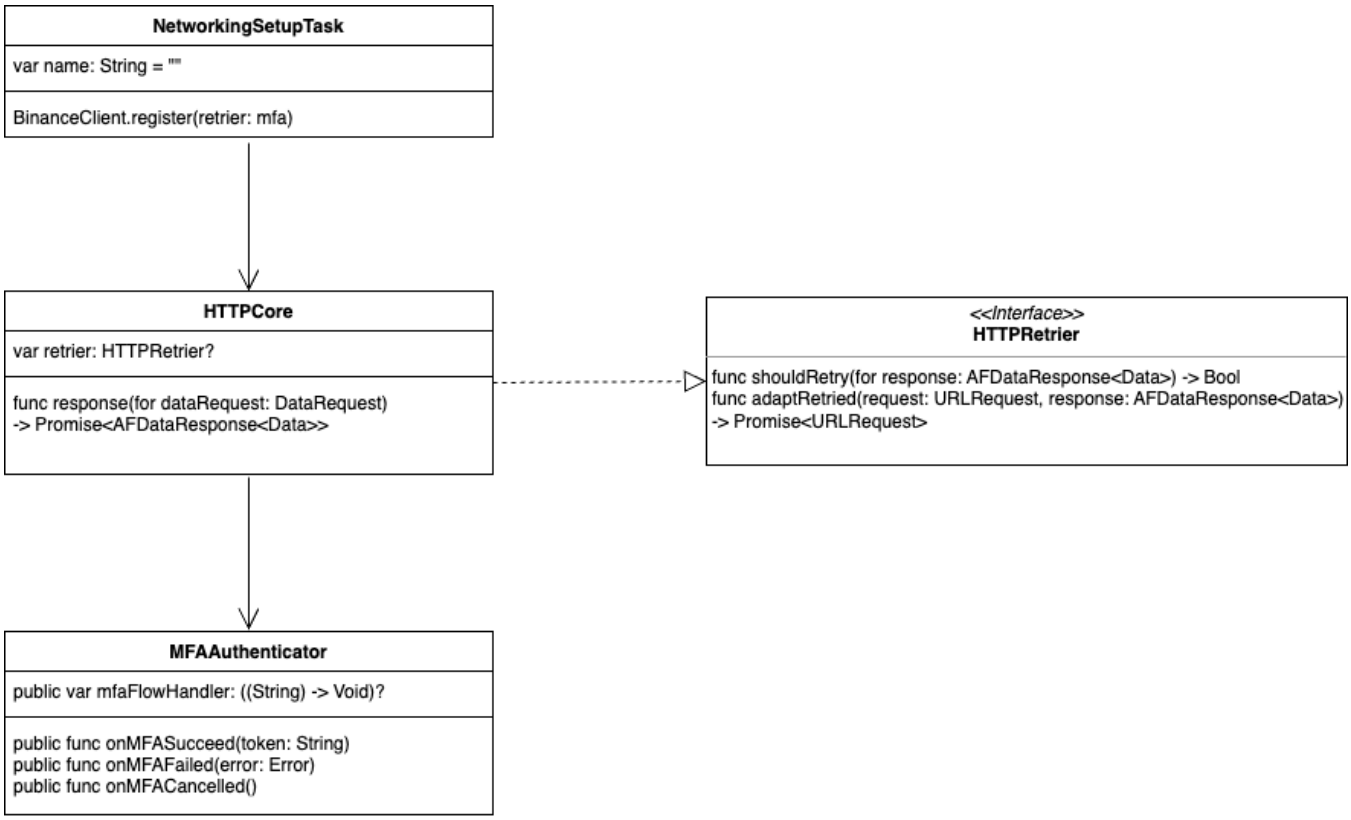
```
let clientId = transferViewController.state.toAccountConfig.baseInfo.clientId
BNCLoadingHUD.showLoading {
    MarginService.transferToThirdWalletMFA(
        clientId: clientId,
        amount: amount,
        asset: asset
    )
}.done { [weak self] in
    guard let self = self else { return }
    let message = "controllers.margin.transferController.success".localized
    let toast = BNCToast(message: message, image: nil, style: .medium)
    self.viewController.view.showToast(view: toast, duration: 1.0) { [weak self] _ in
        guard let self = self else { return }
        self.coordinator?.transferCoordinatorSelectClose(self)
    }
}.catch { error in
    guard let codeError = error as? CodeError else {
        return
    }
    guard codeError.code != Constants.mfaErrorCodeCancel else {
        return
    }
    self.log(message: "third-wallet transfer exempt from MFA failed", error: codeError)
    BNCAPIErrorManager.shared.showError(message: "Third Transfer Error", error: codeError)
}
```

2、Diagram

sequence diagram:



class diagram



2、Core Code

1. Register MFAAuthenticator

```
class NetworkingSetupTask: BNCLaunchTask {

  let mfa = MFAAuthenticator()
  mfa.mfaFlowHandler = { bizNo in
        AppCoordinator.current?.beginMFAAuthentication(bizNo: bizNo, completion: { result in
                switch result {
                case let .success(token):
                    mfa.onMFASucceed(token: token)
                case let .failure(error):
                    mfa.onMFAFailed(error: error)
                }
        })
   }

   BinanceClient.register(retrier: mfa)
}
```

2. HTTPCore: re-request

```
extension HTTPCore {
    private func response(for dataRequest: DataRequest) -> Promise<AFDataResponse<Data>> {
        updateRequestCount(change: .increase)
        return dataRequest.promisedResponseData(queue: queue).then(on: queue) { response ->
Promise<AFDataResponse<Data>> in
            self.updateRequestCount(change: .decrease)
            self.logger?.log(response: response)
            if let retrier = self.retrier, let request = response.request {
                if retrier.shouldRetry(for: response) {
                    return retrier.adaptRetried(request: request, response: response).then(on: self.queue) {
newRequest in
                        self.request(newRequest)
                    }.recover(on: self.queue) { _ in
                        // return first request result if retrier adapts request failed
                        Promise.value(response)
                    }
                }
            }
            return Promise.value(response)
        }
    }
}
```

3. MFAAuthenticator

```
public class MFAAuthenticator: HTTPRetrier {
     public func shouldRetry(for response: AFDataResponse<Data>) -> Bool {
        guard let responseHeaders = response.response?.allHeaderFields else {
            return false
        }
        guard let shouldChallenge = responseHeaders[
            Constants.ResponseHeaderKey.riskChallengeEnableFlow
        ] as? String else {
            return false
        }
        guard let bizNo = responseHeaders[Constants.ResponseHeaderKey.riskChallengeBizNo] as? String else {
            return false
        }
        BinanceLog.log(category: .http, message: "find MFA headers \(shouldChallenge), \(bizNo)")

        return shouldChallenge == "true"
    }

    public func adaptRetried(request: URLRequest, response: AFDataResponse<Data>) -> Promise<URLRequest> {
        guard let bizNo = response.response?.allHeaderFields[Constants.ResponseHeaderKey.riskChallengeBizNo] as?
String
        else {
            return .init(error: AuthError.cancelled)
        }
        return Promise { resolver in
            BinanceLog.log(category: .http, message: "start MFA Flow \(bizNo)")

            let newItem = MFAAuthItem(request: request, resolver: resolver, bizNo: bizNo)
            switch state {
            case .idle:
                state = .authorizing([newItem])
            case let .authorizing(items):
                state = .authorizing(items + [newItem])
            }
            mfaFlowHandler?(bizNo)
        }
    }
}
```

4. MFAPlugin and AppCoordinator+MFA

```
public final class MFAPlugin: ExternalPlugin {
    static let mfaMPAppID = "cH6sYTb5RTJEkpuwrbn7zm"

    public var actions: [ExternalActionDefinitionConvertible] = [
        "private-mfa-set-challenge-token"
    ]

    public func handle(_ action: Action, callback: Callback?) throws {
        let payload = try action.payload.decode(MFAChallengePayload.self)

        DispatchQueue.main.async {
            let appCoordinator = (UIApplication.shared.delegate as? AppDelegate)?.appCoordinator
            appCoordinator?.onMFASetChallengeToken(payload: payload)
        }

        callback?.success()
    }
}

extension AppCoordinator {
    public func beginMFAAuthentication(bizNo: String, completion: @escaping (Swift.Result<String, Error>) ->
Void) {
        mfaAuthenticationCompletion = completion

        let queryString = ["bizNo": bizNo]
            .map { key, value in "\(key)=\(value)" }
            .joined(separator: "&")

        jumpToMp(scheme: .init(
            appID: MFAPlugin.mfaMPAppID,
            query: queryString,
            scheme: .web,
            showStyle: .present,
            showOptions: .dismissAll,
            presentationStyle: .pageSheet
        ))
    }

    func onMFASetChallengeToken(payload: MFAChallengePayload) {
        if payload.status == 0 {
            mfaAuthenticationCompletion?(.success(payload.challengeToken))
        } else {
            mfaAuthenticationCompletion?(.failure(AppMFAAuthError.business(payload.status)))
        }
    }
}
```