**CMA**

**Streamer**

**Feed**

**API**

**DLL / REST**

**Manual**

**de**

**Utilização**

**Histórico das Revisões**

| Versão | Data | Autor | Descrição |
| --- | --- | --- | --- |
| V2.00 | 19/04/2021 | Celso Zanin | Versão 2.0 do CMA Streamer Feed API e REST. |
| V2.01 | 06/05/2021 | Celso Zanin | Ajustes diversos. |
| V2.02 | 05/07/2021 | Celso Zanin | Adicionados informações de compatibilidade .NET Core 2.0+ e .NET 4.6.1+. |
| V2.03 | 08/04/2022 | Celso Zanin | Novas funcionalidades LoadTest e DBFeed + Lista de Acceptors. |

**Sumário:**

[1) Informações Gerais 5](#_Toc100304718)

[1.1) Novidades da versão 2.0 5](#_Toc100304719)

[1.2) Diagrama de Conexões 7](#_Toc100304720)

[1.2.1) API 7](#_Toc100304721)

[1.2.2) REST 8](#_Toc100304722)

[1.3) Diagrama de Sequência 9](#_Toc100304723)

[1.3.1) Diagrama de sequência da API resumida para obter cotações da bolsa 9](#_Toc100304724)

[1.3.2) Diagrama de sequência da API 10](#_Toc100304725)

[1.3.3) Diagrama de sequência do REST resumido para obter cotações da bolsa 11](#_Toc100304726)

[1.4) CMA Streamer Feed 12](#_Toc100304727)

[1.4.1) API 12](#_Toc100304728)

[1.4.2) REST 14](#_Toc100304729)

[1.5) CMA Web Acceptor 15](#_Toc100304730)

[1.6) Infraestrutura 16](#_Toc100304731)

[1.7) Preparação do Ambiente CMA 18](#_Toc100304732)

[1.7.1) CMA Web Acceptor 18](#_Toc100304733)

[1.7.2) CMA Streamer 18](#_Toc100304734)

[1.7.3) CMA OMS 18](#_Toc100304735)

[1.7.4) CMA Streamer Feed API 21](#_Toc100304736)

[1.7.5) CMA Streamer Feed REST 22](#_Toc100304737)

[2) Informações Técnicas do CMA Streamer Feed API 23](#_Toc100304738)

[2.1) Informações da API 23](#_Toc100304739)

[2.1.1) Versionamento e Distribuição 23](#_Toc100304740)

[2.1.2) Configurando Log e Proxy da API 23](#_Toc100304741)

[2.1.3) Configurações da API 24](#_Toc100304742)

[2.1.4) Habilitando modo Trace 24](#_Toc100304743)

[2.1.5) Detalhes da verificação de funcionamento e monitoração da API 25](#_Toc100304744)

[2.1.5.1) HealthCheck 25](#_Toc100304745)

[2.1.5.2) AcceptorCheckSessions 26](#_Toc100304746)

[2.1.5.3) PingRequest 26](#_Toc100304747)

[2.1.5.4) Audit 27](#_Toc100304748)

[2.1.5.5) Instrumentation 27](#_Toc100304749)

[2.1.6) Configurações necessárias e recomendadas para o Acceptor e Streamer 28](#_Toc100304750)

[2.1.7) Configurações do Acceptor para utilizar “*login*” com “*token*” 28](#_Toc100304751)

[2.1.8) Exemplo com fonte utilizando CMA Streamer Feed API 30](#_Toc100304752)

[2.2) Tratamento de Erros e Exceptions da API 39](#_Toc100304753)

[2.3) Instanciando a API 43](#_Toc100304754)

[2.4) Solicitação de Informações de Market Data para API 45](#_Toc100304755)

[2.4.1) Solicitação de Cotações para API (Quotes) 45](#_Toc100304756)

[2.4.2) Solicitação de Lista de Ações/Opções para API (SymbolSearch) 50](#_Toc100304757)

[2.4.3) Solicitação de Gráfico para API (HistoricalData) 52](#_Toc100304758)

[2.4.4) Solicitação de Livro de Ofertas para API (Book) 53](#_Toc100304759)

[2.4.5) Solicitação de Notícias para API (News) 54](#_Toc100304760)

[2.4.6) Solicitação da Íntegra da Notícia para API (News) 55](#_Toc100304761)

[2.4.7) Solicitação de High Lights para API (Highlights) 57](#_Toc100304762)

[2.5) Solicitação de Informações e Transações de Negociação para API 61](#_Toc100304763)

[2.5.1) Solicitação de Lista de Contas OMS para a API (Client) 61](#_Toc100304764)

[2.5.2) Solicitação de Inclusão de Ordem para API (Order ⇨ NewOrderSingleParam) 63](#_Toc100304765)

[2.5.3) Solicitação de Cancelamento de Ordem para API (Order ⇨ OrderCancelParam) 69](#_Toc100304766)

[2.5.4) Solicitação de Alteração de Ordem para API (Order ⇨ OrderReplaceParam) 71](#_Toc100304767)

[2.5.5) Solicitação de Lista de Ordens para API (Order ⇨ OrderListParam) 74](#_Toc100304768)

[2.5.6) Solicitação de Portfólio (Custódia) para API (Portfolio) 80](#_Toc100304769)

[2.5.7) Solicitação das Regras Verifica Cotação 1/2 para API (RuleVerifyQuotes) 83](#_Toc100304770)

[2.5.8) Solicitação do Extrato Financeiro (FinancialStatements) 86](#_Toc100304771)

[2.5.9) Solicitação de Market Symbol Info para a API 89](#_Toc100304772)

[2.5.10) Solicitação de Lista de Mercado e Símbolo para a API 89](#_Toc100304773)

[2.5.11) Solicitação de Exercício de Opção para a API 91](#_Toc100304774)

[2.5) Solicitação Diversas para API 92](#_Toc100304775)

[2.5.1) Solicitação de Load Test para a API 92](#_Toc100304776)

[2.5.2) Solicitação de Provider ⭢ DBFeed para a API 93](#_Toc100304777)

[3) Informações Técnicas do CMA Streamer Feed REST 94](#_Toc100304778)

[3.1) Informações, Configurações e Instalação 94](#_Toc100304779)

[3.1.1) Versionamento e Distribuição 94](#_Toc100304780)

[3.1.2) Configurações 94](#_Toc100304781)

[3.1.3) Authenticate e Token 96](#_Toc100304782)

[3.1.3) Instalando CMA Streamer Feed REST no MS/IIS 98](#_Toc100304783)

[3.1.4) Instalando CMA Streamer Feed REST como executável (EXE) 99](#_Toc100304784)

[3.2) Informações dos Templates 100](#_Toc100304785)

[3.3) Testando o REST via Postman 103](#_Toc100304786)

[4) Detalhes CMA.StreamerFeedAPI.WinForm.Tester 104](#_Toc100304787)

[4.1) Função LoadTest 104](#_Toc100304788)

[4.2) Função Provider ⭢ DBFeed 105](#_Toc100304789)

[4.2.1) Template via Quotes Template 105](#_Toc100304790)

[4.2.2) Template via Delimited File (BDSQL) 107](#_Toc100304791)

[4.2.3) Template via Database 108](#_Toc100304792)

# 1) Informações Gerais

# 1.1) Novidades da versão 2.0

▪ Desenvolvido em C#, usando .NET Standard, tornando a API compatível tanto com projetos desenvolvidos com .NET Framework, quanto com .NET Core incluindo o novo .NET 5.0;

▪ Novas mensagens:

- HighLight = Relatórios Market Data;

- News (Body) = Íntegra de Notícias;

- MarketSymbolInfo = Informações de mercado e pregão do Símbolo;

- ListMarketSymbol = Lista de Símbolos do OMS;

- SendOptionExercise = Exercício de Opções.

▪ Permite acessar cotações nos modelos RealTime, DelayTime, SnapShot (final do dia) e Contribuidor para as mensagens Quotes, News e HistoricalData (configurado no parâmetro “*VendorId*”);

▪ Permite uso do transporte “*Polling”* para atualizações com conceito de “*assinatura*” (ao assinar recebe um snapshot e em seguida apenas as atualizações online) para as mensagens:

- Quotes = Cotações;

- Book = Livro de Ofertas;

- News = Manchetes das Notícias;

- HistoricalData = Pontos para Gráfico.

Observação: Nas mensagens Quotes e HistoricalData permite solicitar atualização apenas dos campos alterados (diferencial), ou seja, nas atualizações somente os campos alterados serão enviados.

▪ Cache na mensagem de Quotes (cotações) da API para otimização das requisições quando utilizado transporte “*Polling”*;

▪ Agora permite múltiplos contextos de usuários por instância, autenticando no Acceptor e guardando as informações do core por instância no cache;

▪ Possibilidade de acesso ao Market Data e Negociação via REST, detalhes:

- Retorno customizado de acordo com o template (modelo) desejado (templates escritos em Fluid Liquid);

- Templates armazenados em base de dados MS/SQL ou MySQL;

- Autenticação via API e autorização via JWT;

- Caches distribuídos para aumentar a performance armazenando os templates e os templates renderizados;

- Possibilidade de obter inúmeros tipos de retorno via REST (JSON, XML, TXT, CSV e HTML);

- Novas mensagens Check, Audit, Ping Request, Instrumentation e Session Instrumentation para Monitoração do ambiente Acceptor/Streamer.

▪ Melhorias no CMA.StreamerFeedAPI.WinForm.Tester com exemplos de utilização para as funções:

- Inicialização da API;

- Quotes (inclusive utilizando polling e atualização diferencial);

- SymbolSearch;

- Book (inclusive utilizando polling);

- HistoricalData (inclusive utilizando polling);

- Highlights;

- News (inclusive com íntegra da notícia);

- ClientList;

- OrderNew;

- OrderReplace;

- OrderCancel;

- OrderList;

- SendOptionExercise;

- Portfolio;

- RuleVerifyQuotes;

- FinancialStatements;

- MarketSymbol;

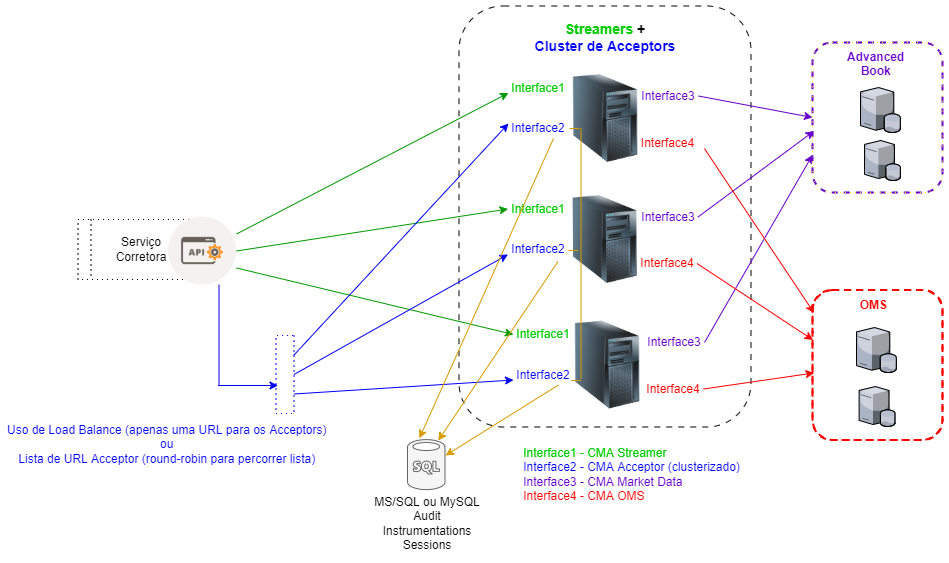
- Monitoramento (HealthCheck, Check, Ping, Audit e Instrumentation);

- LoadTest (solicitações de Inclusão Ordem ou Cotação com mensuração de latências);

- Provider ⭢ DBFeed (simulação do BDSQLQuotes para as bases MS/SQL, MySQL e Oracle).

# 1.2) Diagrama de Conexões

# 1.2.1) API

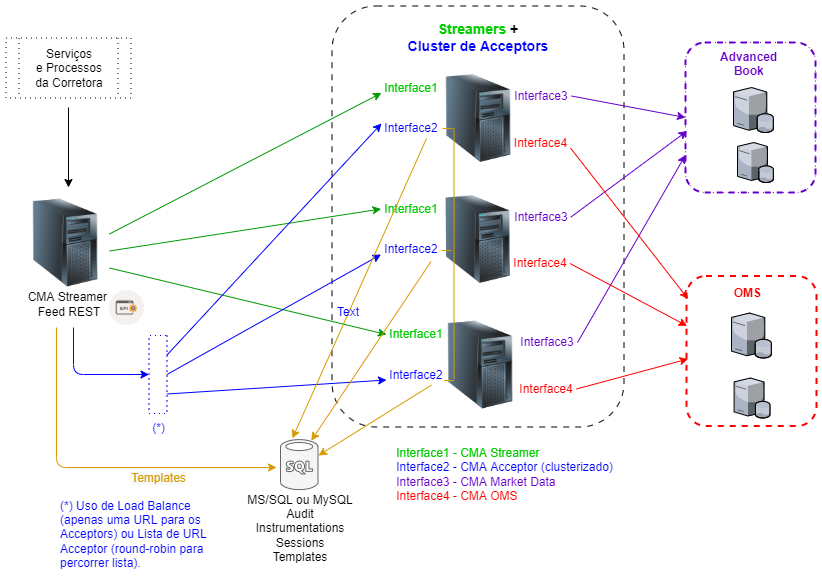


Premissas:

⇨ API Precisa de conectividade com os Acceptors (conexões azuis) e Streamers (conexões verdes);

⇨ Streamers + Cluster de Acceptors precisam de conectividade com os Advanceds e Books (conexões roxas) e OMS (conexões vermelhas).

# 1.2.2) REST



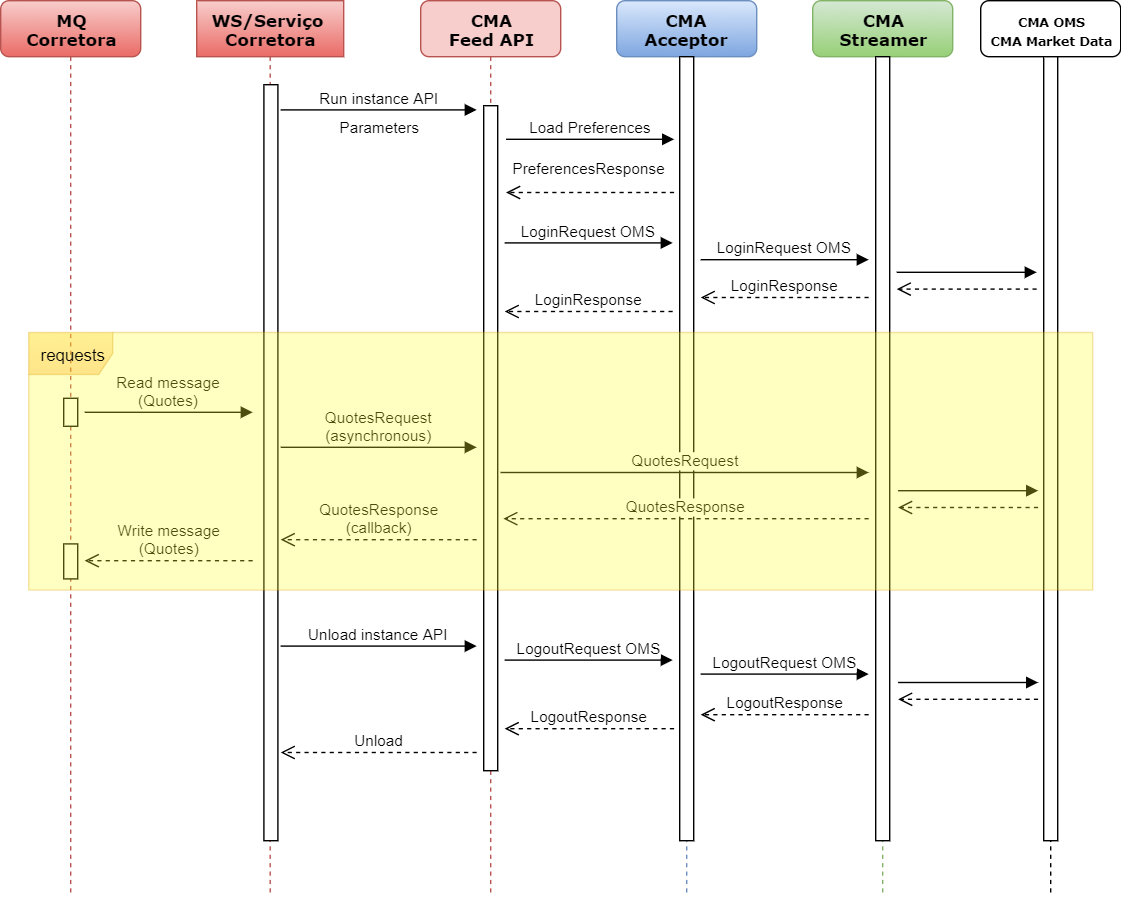
Premissas:

⇨ REST Precisa de conectividade com os Acceptors (conexões azuis), Streamers (conexões verdes) e Base de Dados (conexão amarela);

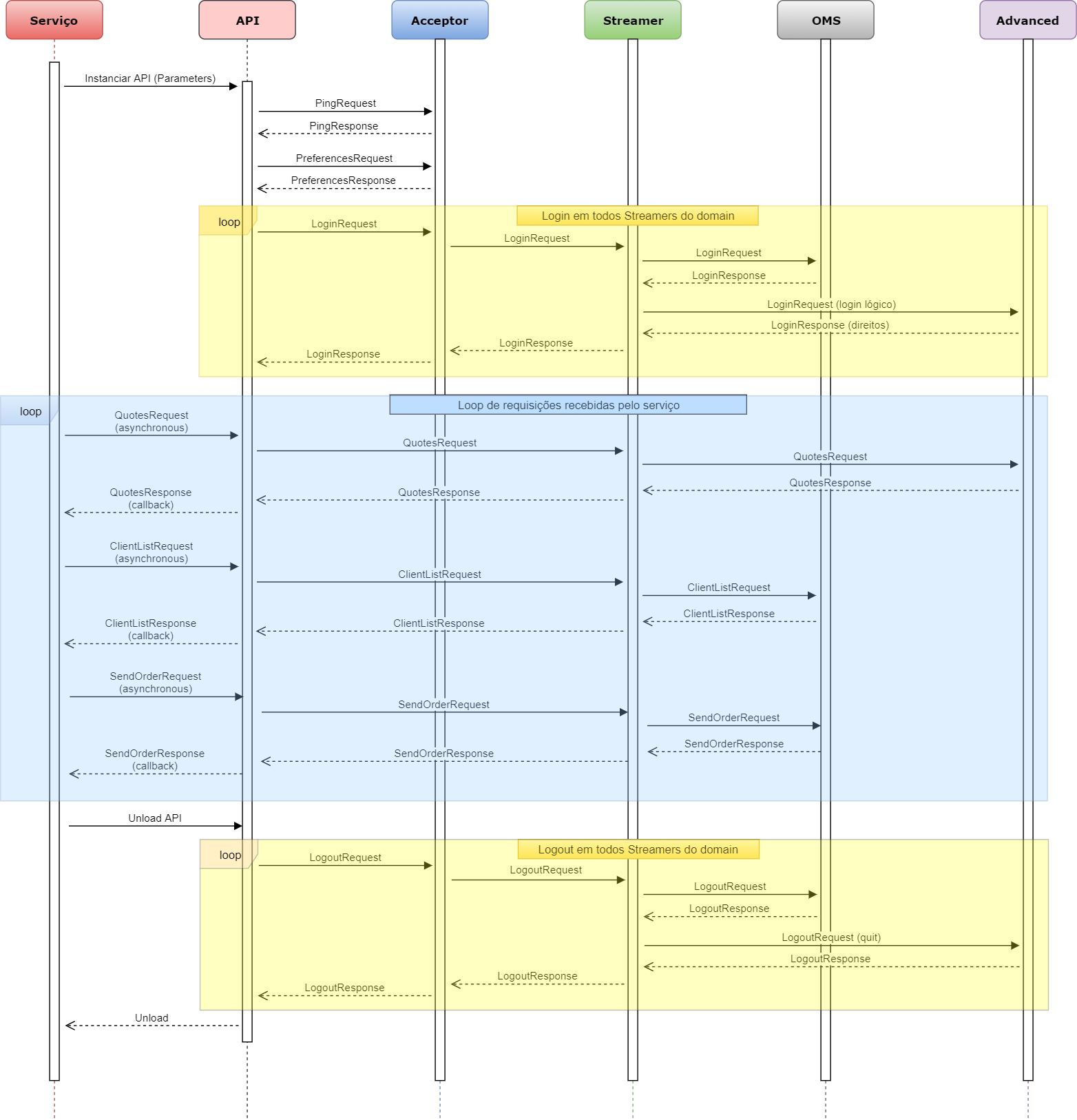
⇨ Streamers + Cluster de Acceptors precisam de conectividade com os Advanceds e Books (conexões roxas) e OMS (conexões vermelhas).

# 1.3) Diagrama de Sequência

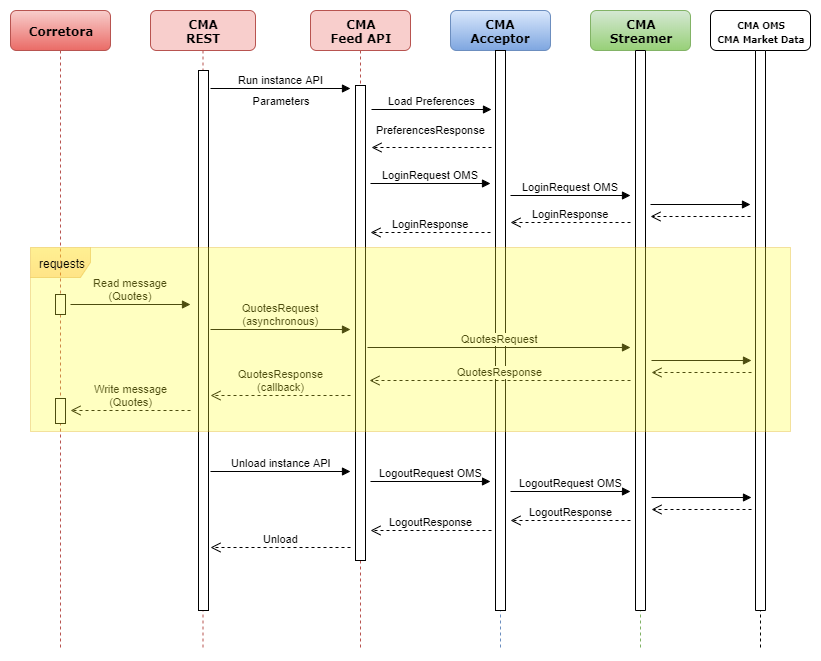
# 1.3.1) Diagrama de sequência da API resumida para obter cotações da bolsa



# 1.3.2) Diagrama de sequência da API



# 1.3.3) Diagrama de sequência do REST resumido para obter cotações da bolsa



# 1.4) CMA Streamer Feed

# 1.4.1) API

▪ API disponível em DLL (biblioteca de classes);

▪ Disponível na versão .NET Standard 2.0 com suporte para as versões:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

<https://dotnet.microsoft.com/en-us/platform/dotnet-standard>

▪ Distribuição via NuGet (pacote CMA.StreamerFeed.Standard.API.2.0.<release>.nupkg);

▪ Objetivo desta API é disponibilizar funções de Market Data (via CMA Market Data) e Negociação (via CMA OMS);

▪ Funções de Market Data:

⇨ **Quotes =** Solicitar cotações de uma lista de ações/opções;

⇨ **SymbolSearch =** Solicitar lista de ações/opções de acordo com filtro;

⇨ **Book =** Solicitar livro de ofertas de uma ação/opção;

⇨ **HistoricalData =** Solicitar pontos do gráfico (períodos de 1min, 5min, 1hora, 1dia, 5dias, Weekly, Monthly, Yearly, etc) para uma ação/opção;

⇨ **Highlights =** Solicitar relatórios:

🢞 Maiores Compradores/Vendedores por Contrato;

🢞 Ativos Mais Negociados por Agente;

🢞 Negócios por Agente;

🢞 Negócios por Agente Acumulado;

🢞 Negócios Cruzados Agente X Contrato;

🢞 Maiores Altas;

🢞 Maiores Baixas;

🢞 Mais Negociadas;

🢞 Resumo de Negócios por Preço Médio.

⇨ **News =** Solicitar notícias.

⇨ **News – Boby =** Solicitar íntegra de notícia.

Observação: Utilizar login do Market Data configurando "*Service=m*" (lembrando que ao acessar Negociação também é liberado Market Data).

▪ Funções de Negociação Renda Variável (via OMS):

⇨ **Client – ClientList** = Envio da lista de clientes dado um determinado filtro;

⇨ **Order – NewOrderSingle** = Envio de ordem;

⇨ **Order – OrderReplace** = Solicitação de alteração de ordem;

⇨ **Order – OrderCancel** = Solicitação de cancelamento de ordem;

⇨ **Order – OrderList** = Solicitação de lista de ordens;

⇨ **SendOptionExercise** = Solicitação de exercício de opção;

⇨ **Portfolio** = Solicitar portfólio (custódia);

⇨ **RuleVerifyQuotes** = Solicitar túnel de preços;

⇨ **FinancialStatements** = Solicitar extrato financeiro;

⇨ **MarketSymbolInfo =** Informações de mercado e pregão do símbolo;

⇨ **ListMarketSymbol =** Lista dos principais símbolos do mercado.

Observação: Utilizar login do OMS configurando "*Service=o"*.

# 1.4.2) REST

▪ Objetivo do REST é disponibilizar funções de Market Data (via CMA Market Data) e Negociação (via CMA OMS);

▪ Funções de Market Data:

⇨ **Quotes =** Solicitar cotações de uma lista de ações/opções;

⇨ **Symbol =** Solicitar lista de ações/opções de acordo com filtro;

⇨ **Book =** Solicitar livro de ofertas de uma ação/opção;

⇨ **HistoricalData =** Solicitar pontos do gráfico (períodos de 1min, 5min, 1hora, 1dia, 5dias, Weekly, Monthly, Yearly, etc) para uma ação/opção;

⇨ **Highlights =** Solicitar relatórios:

🢞 Maiores Compradores/Vendedores por Contrato;

🢞 Ativos Mais Negociados por Agente;

🢞 Negócios por Agente;

🢞 Negócios por Agente Acumulado;

🢞 Negócios Cruzados Agente X Contrato;

🢞 Maiores Altas;

🢞 Maiores Baixas;

🢞 Mais Negociadas;

🢞 Resumo de Negócios por Preço Médio.

⇨ **News =** Solicitar notícias;

⇨ **News/Boby =** Solicitar íntegra de notícia.

Observação: Utilizar login do Market Data configurando "*Service=m*" (lembrando que ao acessar Negociação também é liberado Market Data).

▪ Funções de Negociação de Renda Variável:

⇨ **OrderSingle =** Inclusão de Ordem;

⇨ **Orderreplace =** Alteração de Ordem;

⇨ **OrderCancel =** Cancelamento de Ordem;

⇨ **OrderList =** Lista de Ordens;

⇨ **Porffolio =** Custódia.

Observação: Utilizar login do OMS configurando "*Service=o*".

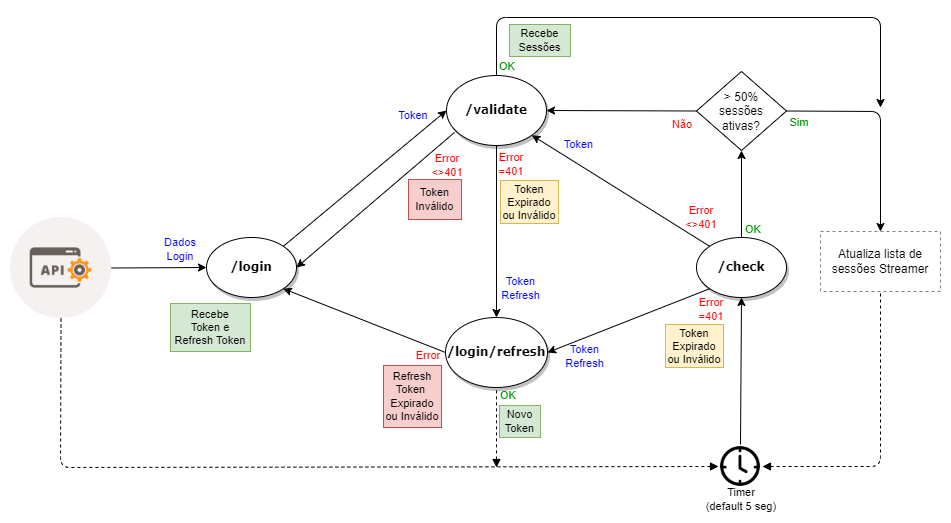
# 1.5) CMA Web Acceptor

▪ Implementa *'Single Sign-On*' (SSO) para as conexões com os Streamers;

▪ Efetuará a lógica de autenticação no POOL (SHARD) de Streamers;

▪ Será o responsável em criar as sessões nos Streamers, mantê-las online até sua finalização seja por ‘*timeout’* ou ‘*logout’* e orquestração do Shard de Streamers dedicados ao atendimento dos serviços da API;

▪ Diagrama com detalhes do fluxo de validação executados pela API com o Acceptor:



⇨ **login** = Função executada para obter Token de segurança JWT com credenciais de autenticação;

⇨ **validate** = Função executada para obter as sessões do Streamer (número de sessões configurada no Acceptor parâmetro “*shard*”);

⇨ **check** = Função executada para verificar o estado das sessões do Streamer, sendo que esta função deve ser chamada a cada “n” segundos (recomendamos intervalo de 5 segundos);

⇨ **login/refresh** = Função executada para revalidar o Token no Acceptor (tempo configurado no Acceptor parâmetro “*token\_ttl*”).

Onde:

- Token = Token para acesso ao Acceptor (tempo de validade configurado em “*token\_ttl*”);

- Token Refresh = Token para efetuar revalidação do Token original (tempo determinado na configuração “*refresh\_token\_ttl*”).

# 1.6) Infraestrutura

1.6.1) Hardware e software recomendado para o CMA Streamer Feed API:

▪ Mesmo hardware e software recomendado para o Serviço Corretora, porém recomendamos:

⇨ 8 x CPU clock 2,4GHz+;

⇨ 8GB RAM;

⇨ 80GB Disco;

⇨ Windows 2012 R2 64-bit;

⇨ .NET runtime 4.6.1+.

▪ Interfaces de Rede:

⇨ Rede dedicada para comunicação entre Streamer Feed API e Acceptor;

⇨ Rede dedicada para comunicação entre Streamer Feed API e Streamer.

Observação: Utilizamos 2 interfaces de rede para otimizar a utilização da placa de rede e permitir maior controle e segurança, porém a corretora pode adotar outras configurações.

1.6.2) Hardware e software recomendado para o CMA Streamer e CMA Web Acceptor Instalados no mesmo servidor):

▪ 3 (três) servidores físicos ou virtuais (VM):

⇨ 8 x CPU clock 2,4GHz+;

⇨ 16GB RAM;

⇨ 300GB Disco;

⇨ Windows 2012 R2 64-bit;

⇨ Java JDK 8 64-bit;

⇨ Acesso a base de dados MS/SQL 2012 64-bit.

▪ Interfaces de Rede:

⇨ Rede dedicada para comunicação entre Acceptor e Streamer Feed API;

⇨ Rede dedicada para comunicação entre Streamer e Streamer Feed API;

⇨ Rede dedicada para comunicação entre Streamer e Market Data;

⇨ Rede dedicada para comunicação entre Streamer e OMS.

Observação: Utilizamos 4 interfaces de rede em cada hardware para otimizar a utilização da placa de rede e permitir maior controle e segurança, porém a corretora pode adotar outras configurações.

▪ O acesso ao CMA Web Acceptor deverá utilizar o “*load balance*” para permitir contingenciamento e acesso através de um único endereço, ou ainda, uma lista de Acceptors para a API percorrer usando “*roun-robin*” (Acceptors do mesmo cluster);

▪ Estimativa de hardware baseado nas premissas:

⇨ Ativos: Até 50 ativos simultâneos (somente BOVESPA);

⇨ Acessos simultâneos: 5.000 a 10.000 clientes acessando os serviços da Corretora;

⇨ Capacidade: 1.000 ordens por dia [10:00 – 18:00] (200 ordens por hora ou 5 ordens por minuto).

1.6.3) Banco de dados para o CMA Web Acceptor:

▪ Banco de dados MS/SQL Server versão 2012 ou superior;

▪ A base de dados será utilizada para armazenamento de auditoria de acessos.

1.6.4) Conexões necessárias:

▪ CMA Streamer Feed API:

⇨ Conexão com o CMA Web Acceptor;

⇨ Conexão com o CMA Streamer.

▪ CMA Web Acceptor:

⇨ Conexão com o CMA Streamer;

⇨ Conexão com os demais CMA Web Acceptors;

⇨ Conexão com o servidor MS/SQL (base de dados do Acceptor para auditoria).

▪ CMA Streamer:

⇨ Conexão com o CMA Market Data (servidores de cotações, notícias, gráficos e livros de ofertas);

⇨ Conexão com o CMA OMS (servidor de negociação Bovespa módulo GO).

Observação: Vide item 1.2 (Diagrama de conexões da API).

# 1.7) Preparação do Ambiente CMA

# 1.7.1) CMA Web Acceptor

▪ Configurações no arquivo “*acceptor.conf*” do diretório “*C:\CMA\Acceptor\1.n.n.n\conf*” (exemplo: 1.0.27.0);

▪ Parâmetro “*acceptor.domains*”, host-name e porta são informações necessárias para instanciar a API, exemplos:

⇨ InstanceParams.AcceptorUrl = "http://dev.cma.com.br:9080/acceptor";

⇨ InstanceParams.DomainId = "streamerfeedapi".

▪ Conexões de rede necessárias:

⇨ Acesso ao Streamer (ip/host-name e porta);

⇨ Acesso ao Advanced (Market Data) (ip/host-name e porta);

⇨ Banco de dados MS/SQL para armazenamento do log de auditoria.

# 1.7.2) CMA Streamer

▪ Configurar os logins (físicos) do Advanced;

▪ Logins (físicos) do Advanced devem ter direito a origens utilizadas pela API/REST;

▪ Conexões de rede necessárias:

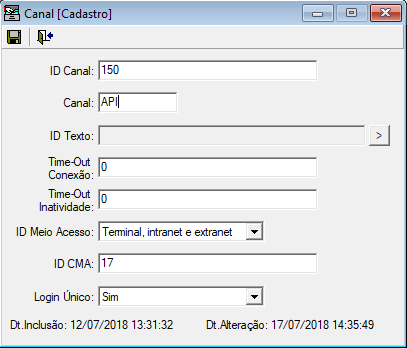
⇨ Acesso ao Advanced (Market Data) com informações ip, host-name e porta;

⇨ Acesso ao OMS (negociação módulo GO) com informações ip, host-name e porta.

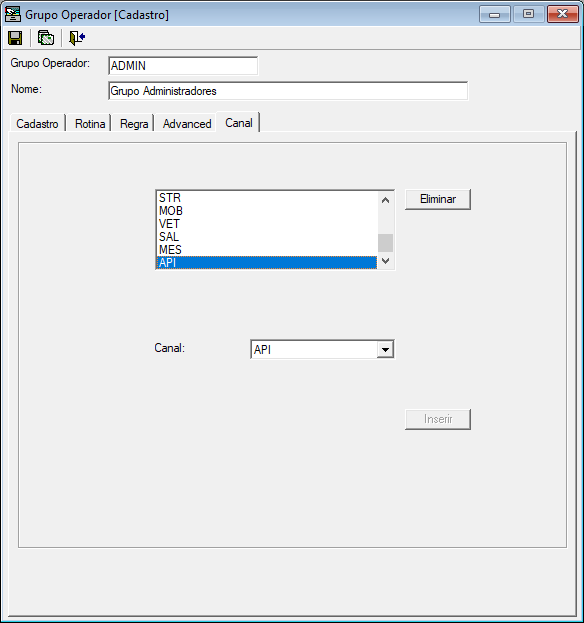
▪ Verificar configuração adequada para a utilização da assinatura eletrônica do OMS.

# 1.7.3) CMA OMS

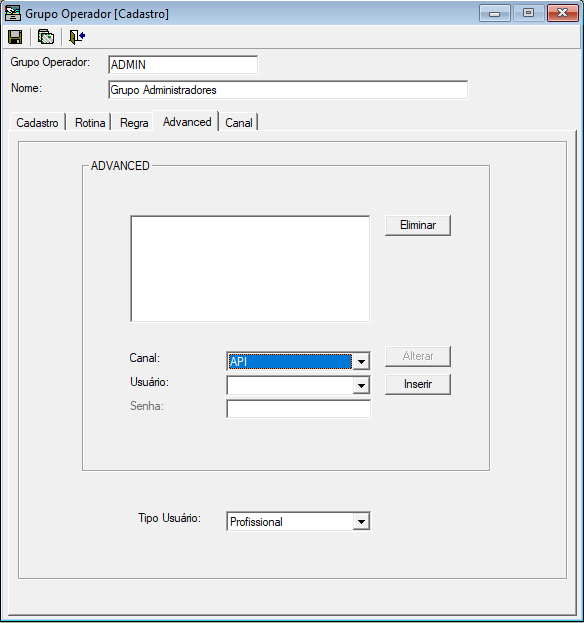
▪ Cadastrar o canal “*API*” (ID=150) via módulo AC (menu Administração - Sistema - Aplicação - Canal):



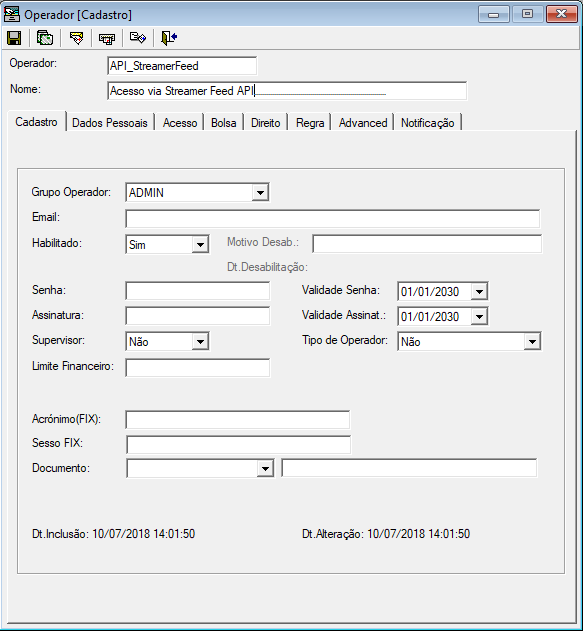
▪ Habilitar o canal “*API*” via módulo AC no Grupo Operador (menu Cadastro - Operador - Grupo Operador):



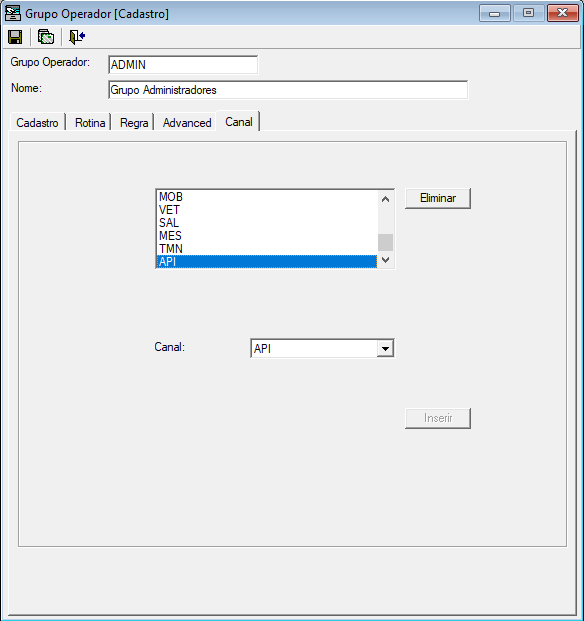
▪ Cadastrar usuário Advanced para o canal “*API*” no Grupo Operador via módulo AC (menu Cadastro - Operador - Grupo Operador):



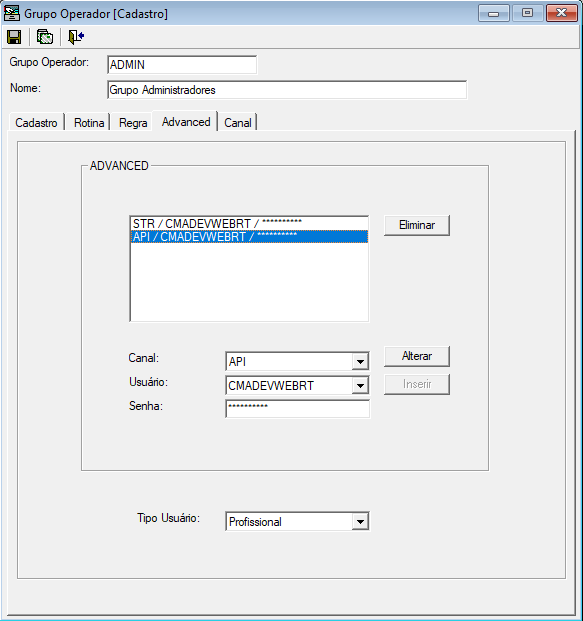
▪ Cadastrar o(s) operador(es) que serão enviados para a API via módulo AC (menu Cadastro - Operador - Operador):



⇨ Habilitar o canal API (150) para o Grupo do Operador:



⇨ Cadastrar o usuário Advanced no Grupo Operador ou Operador que deverá ser utilizado pelo canal API, exemplo no Grupo Operador:



⇨ Recomendamos atribuir mnemônico do operador que identifique rapidamente a API e seu respectivo ambiente (exemplo: API\_StreamerFeed-DEV para o ambiente de desenvolvimento);

⇨ A estratégia de utilizar um, ou mais, códigos de operadores será determinado pela Corretora (deve estar de acordo com controle de acessos simultâneos);

⇨ Atribuir senha e colocar a data de validade da senha distante (exemplo: 31/12/2030);

⇨ Adicionar as configurações de Bolsa, Direito (acesso irrestrito) e Regra, conforme determinação da Corretora;

⇨ A regra de Suitability não será aplicada pelo OMS para o canal “*API*” (observação: se utilizado outro canal a regra do Suitability deve ser implementada);

⇨ Observar a configuração da necessidade da “*assinatura*”, se configurada no Streamer (oms\_signature\*) a API deverá enviar a assinatura nas operações com o OMS;

⇨A “*dupla autenticação*” deve ser desligada na configuração do Streamer (oms\_double\_login = 0);

⇨ Mnemônico e a senha do Operador são informações necessárias para instanciar a API, exemplo:

⬝ InstanceParams.User = API\_StreamerFeed;

⬝ InstanceParams.Password = 123456.

# 1.7.4) CMA Streamer Feed API

▪ Informações necessárias para instanciar a API:

⇨ AcceptorUrl = URL(s) do CMA Web Acceptor(s);

⇨ DomainId = Domain de Streamers;

⇨ User e Password:

⬝ OMS: Mnemônico e Senha do Operador CMA OMS (vide item 1.7.3);

⬝ *Ou* MarketData: Mnemônico e Senha do MarketData (CMA Advanced).

⇨ Dados do LoginToken quando acesso via TOKEN.

▪ Conexões de rede necessárias:

⇨ Acesso ao Streamer (ip/host-name e porta);

⇨ Acesso ao Acceptor via “*load balance*” (ip/host-name e porta).

⇨ Acesso aos Acceptors quando utilizado lista (ip/host-name e porta).

# 1.7.5) CMA Streamer Feed REST

▪ Informações necessárias o REST:

⇨ AcceptorUrl = URL(s) do CMA Web Acceptor(s);

⇨ DomainId = Domain de Streamers;

⇨ User e Password:

⬝ OMS: Mnemônico e Senha do Operador CMA OMS (vide item 1.7.3);

⬝ *Ou* MarketData: Mnemônico e Senha do MarketData (CMA Advanced).

⇨ Dados do LoginToken quando acesso via TOKEN.

▪ Conexões de rede necessárias:

⇨ Acesso ao banco de dados para armazenamento dos templates;

⇨ Acesso ao Streamer (ip/host-name e porta);

⇨ Acesso ao Acceptor quando utilizado “*load balance*” (ip/host-name e porta)

⇨ Acesso aos Acceptors quando utilizado lista (ip/host-name e porta).

# 2) Informações Técnicas do CMA Streamer Feed API

# 2.1) Informações da API

# 2.1.1) Versionamento e Distribuição

▪ Versionamento no padrão A.B.C.F “*major.minor.branch.fix-release*”:

⇨ A - major = Grandes modificações e mudanças e quebra de compatibilidade;

⇨ B - minor = Novas funcionalidades mantendo compatibilidade;

⇨ C - branch = Ramificação com novas funcionalidades e ajustes mantendo compatibilidade;

⇨ F - fix-release = Correções de falhas e ajustes mantendo compatibilidade.

▪ CMA Streamer Feed API (CMA.StreamerFeed.Standard.API):

⇨ Distribuição via NuGet (gerenciador de bibliotecas para a plataforma .NET);

⇨ Arquivo ZIP com extensão “*.nupkg*” que contém o código compilado da API (DLL);

⇨ Também contém informações das dependências utilizadas (exemplo: AKKA e Sirilog);

⇨ Exemplo: CMA.StreamerFeed.Standard.API.2.0.0.520.nupkg;

⇨ Disponível nas versões **.NET Standard 2.0**;

⇨ Compatível com .NET Framework 4.6.1 (ou superior), .NET 5.0 e .NET Core 2.0 (ou superior);

⇨ Recomendamos o uso dos ambientes de desenvolvimento (IDE) Visual Studio 2019 Community Ou JetBrains Rider;

⇨ Instalando o pacote NuGet da API no Visual Studio vide <https://docs.microsoft.com/en-us/nuget/quickstart/install-and-use-a-package-in-visual-studio> (VS 2019 – Ferramentas – Gerenciador de Pacotes NuGet).

▪ CMA Streamer Feed REST (CMA.StreamerFeed.REST):

⇨ Distribuição via arquivos dispostos em diretório (para instalação no IIS ou execução via EXE);

⇨ Detalhes no item 3.

# 2.1.2) Configurando Log e Proxy da API

▪ Configuração no arquivo “*App.config*” do serviço que está utilizando a DLL;

▪ Configurações:

<?xml version="1.0" encoding="utf-8"?>

<configuration>

...

<appSettings>

...

<add key="CMA.StreamerFeed.Api:serilog:minimum-level" value="Warning" />

<add key="CMA.StreamerFeed.Api:serilog:using:File" value="Serilog.Sinks.File" />

<add key="CMA.StreamerFeed.Api:serilog:write-to:File.path" value=".\log\CMAStreamerFeed.log"/>

<add key="CMA.StreamerFeed.Api:serilog:write-to:File.rollingInterval" value="Day" />

<add key="CMA.StreamerFeed.Api:serilog:write-to:File.shared" value="true" />

<add key="CMA.StreamerFeed.Api:proxy:url" value="" />

<add key="CMA.StreamerFeed.Api:proxy:user" value="" />

<add key="CMA.StreamerFeed.Api:proxy:password" value="" />

<add key="CMA.StreamerFeed.Api:Flurl:AcceptEncoding" value="gzip, deflate, br" />

...

</appSettings>

...

</configuration>

▪ serilog:minimum-level = Níveis de configuração do Serilog:

⇨ Verbose = Informações de rastreamento e minúcias de depuração; geralmente apenas ligado em situações incomuns (Trace será gravado no log - vide item 2.1.3.) ;

⇨ Debug = Fluxo de controle interno e dumps de e diagnóstico para facilitar a identificação de problemas reconhecidos;

⇨ Information = Eventos de interesse ou que tenham relevância para observadores externos; o nível de registro mínimo ativado padrão;

⇨ Warning = Indicadores de possíveis problemas ou degradação de serviço / funcionalidade (recomendado para o ambiente de produção);

⇨ Error = Indicando uma falha no aplicativo ou no sistema conectado;

⇨ Fatal = Erros críticos que causam falha completa do aplicativo.

▪ serilog:using:File = Tipo do arquivo ("*Serilog.Sinks.File*");

▪ serilog:write-to:File.path = Local e nome do arquivo de log (recomendamos "*.\log\CMA.StreamerFeed.log*" e caso instalado no IIS-Global.asax o “*path*” deve ser completo);

▪ serilog:write-to:File.rollingInterval = Intervalo para rotação (recomendamos "*Day*" para gerar arquivos diários);

▪ serilog:write-to:File.shared = Deve ser “*true*” (permitindo escrita compartilhada);

▪ proxy:url = URL para acesso ao servidor de Proxy (caso não informado vai utilizar conexão direta sem uso do Proxy), exemplo: “*http:/10.0.28.1:8080*”;

▪ proxy:user = Usuário para login no servidor de Proxy (caso não informado não é necessário user/pwd para utilizar o Proxy);

▪ proxy:password = Senha para login no servidor de Proxy;

▪ flurl:AcceptEncoding = Configuração “*Accept Encoding request HTTP*” que permite utilizar compressão (exemplo: “*gzip, deflate, br*”) nas comunicações http(s) com o Acceptor e Streamer.

# 2.1.3) Configurações da API

▪ Configuração no arquivo “*App.config*” do serviço que está utilizando a DLL;

▪ Configurações:

<?xml version="1.0" encoding="utf-8"?>

<configuration>

...

<appSettings>

...

<add key="TimerCheckAcceptor" value="5" />

<add key="TimerCheckAcceptorWithLoginProblem" value="30" />

<add key="PercentageActiveSessions" value="50" />

...

</appSettings>

...

</configuration>

▪ Onde:

- TimerCheckAcceptor = Intervalo de tempo (em segundos) para verificação das sessões junto ao Acceptor (função “*check*”), se não informado a API vai utilizar 5 segundos;

- TimerCheckAcceptorWithLoginProblem = Intervalo de tempo (em segundos) após um “*login*” recusado pelo Advanced ou OMS, se não informado a API vai utilizar 30 segundos;

- PercentageActiveSessions = Percentual de sessões ativas aceitas antes da solicitação de novas sessões (função “*validate*”), se não informado a API vai utilizar 50%.

# 2.1.4) Habilitando modo Trace

▪ Através do Trace é possível extrair a pilha de execuções processadas dentro da API, permitindo assim efetuar depurações e análises diversas;

▪ Todas as solicitações permitem habilitar o retorno do Trace;

▪ Para configurar o Trace fazer “*EnableTrace = true*” (exemplo: quotesParam.EnableTrace = true);

▪ Toda mensagem de resposta (resultado de saída em “*\*Result*”) será enviado o Trace com a estrutura:

⇨ Trace

- TotalLatency (string) = Latência de execução da API (em ms);

- (List) TraceItems:

- ReceivedObjectJson (string) = Requisição recebida no formato JSON;

- Self (string) = Classe/Ator atual;

- Sender (string) = Classe/Ator que será chamado;

- SentObjectJson (string) = Retorno no formato JSON;

- TimeStamp (System.DateTime) = Data/hora na criação do Trace.

Observação: Se não habilitado o Trace “*null*” será retornado.

▪ Se “*serilog:minimum-level = Verbose*” o Trace será gravado no log (independente se solicitado “*EnableTrace = true*”).

# 2.1.5) Detalhes da verificação de funcionamento e monitoração da API

# 2.1.5.1) HealthCheck

▪ Método **HealthCheck**;

▪ Parâmetros de entrada = fullHealthCheck (bool):

⇨ Se false = Sintético (recomendado);

⇨ Se true = Analítico com lista de Sessões do Streamer .

▪ Retorno = Estrutura **HealthCheck**:

⇨ Api = Informações da API:

⬝ InstanceName (string) = Nome da instância da API;

⬝ TimeStamp (numeric) = Timestamp para controle de travamento (a cada chamada deve estar diferente);

⬝ Status (bool) = Se True API está funcionando, caso False API está com problemas;

⬝ StatusInformation = Estatísticas das conexões:

- StreamersQtt (int) = Quantidade de Sessões do Streamer;

- AcceptorsQtt (int) = Quantidade de conexões com o Acceptor.

⇨ Msg = Informações do Acceptor (enviado quando analítico):

⬝ success (bool) = Se True Acceptor retornou a lista de sessões;

⬝ code (int) = Código da chamada “*http*” do Acceptor;

⬝ msg = Detalhes das sessões com o Streamer:

- expirationTime (bool) = Data no formato “*AAAA-MM-DD HH:MM:SS.III*”;

- uuid (string) = Sessão do Acceptor;

- shard (bool) = Número de conexões com o “*pool*” Streamer;

- timestamp (numeric) = TimeStamp;

- streamerSessionIds = Lista de sessões do Streamer:

- sessionId (string) = Sessão do Streamer;

- streamerId (string) = Nome da instância do Streamer;

- errorMsg (string) = Mensagem de erro;

- errorStatus (string) = Status do erro;

- valid (bool) = Se True sessão está válida, caso False sessão está inválida.

Observações:

⇨ Esta funcionalidade informa se a API está com conexões válidas com o Acceptor e Streamer (monitorações devem ser executadas nestes produtos);

⇨ O campo “*Api.Status*” indica se a API está funcional e conectada com Acceptor e Streamer.

▪ Exemplo de chamada do HealthCheck:

using System;

using CMA.StreamerFeed.Models.Interfaces;

...

// Solicitar Health Check (Sintético)

var result = cmaStreamerFeed.HealthCheck(false).Result;

Console.WriteLine($"\Health Check = " + result.ToJSON());

...

# 2.1.5.2) AcceptorCheckSessions

▪ Método **AcceptorCheckSessions**;

▪ Parâmetro de entrada = Nenhum (retorna todas sessões).

▪ Retorno = Estrutura **AcceptorCheckResponse**:

⇨ success (bool) = Se True Acceptor retornou a lista de sessões;

⇨ code (int) = Código da chamada “*http*” do Acceptor;

⇨ msg = Detalhes das sessões com o Streamer:

⬝ expirationTime (bool) = Data no formato “*AAAA-MM-DD HH:MM:SS.III*”;

⬝ uuid (string) = Sessão do Acceptor;

⬝ shard (bool) = Número de conexões com o “*pool*” Streamer;

⬝ timestamp (numeric) = TimeStamp;

⬝ streamerSessionIds = Lista de sessões do Streamer:

- sessionId (string) = Sessão do Streamer;

- streamerId (string) = Nome da instância do Streamer;

- errorMsg (string) = Mensagem de erro;

- errorStatus (string) = Status do erro;

- valid (bool) = Se True sessão está válida, caso False sessão está inválida.

⇨ errors (array string) = Lista de erros.

Observações: Esta funcionalidade informa as conexões com Streamer e sua situação.

# 2.1.5.3) PingRequest

▪ Método **PingRequest**;

▪ Parâmetro de entrada = Nenhum;

▪ Retorno = Estrutura **PingResponse**:

⇨ success (bool) = Identifica se a chamada foi realizada com sucesso;

⇨ status (bool) = Código do Status (0 indica sucesso);

⇨ acceptorId (string) = Id do Acceptor;

⇨ errors (array string) = Lista de erros;

⇨ code (int) = Código da chamada “*http/https*” do Acceptor (200 indica sucesso);

⇨ msg = Detalhes do Acceptor:

⬝ acceptorId (string) = Id do Acceptor;

⬝ timestamp (numeric) = TimeStamp.

Observações: Esta funcionalidade executa um ping no Acceptor (verifica se a conectividade e o próprio Acceptor está funcional).

# 2.1.5.4) Audit

▪ Método **Audit**;

▪ Parâmetros de entrada:

⇨ User (string) = Usuário para acesso a trilha de auditoria do Acceptor;

⇨ Password (string) = Senha do usuário.

▪ Retorno = Estrutura **AuditResponse**:

⇨ instanceName (string) = Nome da instância do Acceptor;

⇨ streamerSessionId (string) = Id da Sessão do Streamer;

⇨ streamerKey (string) = Código do Streamer;

⇨ ip (string) = Ip do solicitante;

⇨ channel (string) = Canal do solicitante;

⇨ userAgent (string) = “*User Agente*” (cabeçario do “*http*”) do solicitante;

⇨ cmaUser (string) = Usuário do solicitante;

⇨ sessionId (string) = Id da Sessão do solicitante;

⇨ type (string) = Tipo de validação;

⇨ createdAt (string) = Data de criação (YYYY-MM-DD HH:MM:SS:NNN);

⇨ customerId (string) = Id do Cliente;

⇨ text (string) = Texto do erro quando requisição negada (status = FAIL);

⇨ status (string) = SUCCESS indica requisição atendida e FAIL requisição não atendida.

Observações: Esta funcionalidade lista todos os comandos (solicitações) executados no Acceptor (trilha de auditoria).

# 2.1.5.5) Instrumentation

▪ Método **Instrumentation**;

▪ Parâmetros de entrada estrutura **InstrumentationParam**:

⇨ AcceptorUrl [obrigatório] (string) = URL(s) do CMA Web Acceptor(s);

⇨ Domain [obrigatório] (string) = Domain de Streamers;

⇨ Username [obrigatório] (string) = Usuário para acesso a instrumentação do Streamer;

⇨ Password [obrigatório] (string) = Senha do usuário;

⇨ Class [obrigatório] (string) = Classe executada;

⇨ Icase [obrigatório] (bool) = Indica se o filtro de “*Class*” deve usar “*case sensitive*”;

⇨ KeepSessionAlive [obrigatório] (bool) = Filtro para

⇨ ClientIp [opcional] (string) = Filtro para Ip;

⇨ StreamerId [obrigatório] (string) = Filtro para Id do Streamer.

▪ Retorno = Estrutura **InstrumentationResponse**:

⇨ classes:

⬝ uname (string) = Instrumento;

⬝ data (string) = Contador.

⇨ sessionIdAlias (string) = Id da Sessão;

⇨ clientIp (string) = Ip do cliente;

⇨ id (integer) = Id do instrumento;

⇨ name (string) = InstrumentationResponse;

⇨ sessionId(string) = Session Id;

⇨ status (bool) = Código do Status (0 indica sucesso);

⇨ success (bool) = Identifica se a chamada foi realizada com sucesso;

⇨ sync (bool) = Síncrono;

⇨ textual (string) = Texto do instrumento;

⇨ type (string) = Tipo do instrumento;

⇨ version (integer) = Id da versão;

⇨ uid (string) = User Id;

⇨ msg (string) = Mensagem.

Observações: Esta funcionalidade lista informações de instrumentação do CMA Advanced Server (informações de métricas de vários pontos de processamento).

# 2.1.6) Configurações necessárias e recomendadas para o Acceptor e Streamer

▪ Acceptor:

acceptor.domain.streamerfeedapi.session\_ttl = **300**

# Validade da sessão do Acceptor (em segundos)

acceptor.domain.streamerfeedapi.token\_ttl = **900**

# Validade do Token (em segundos)

acceptor.domain.streamerfeedapi.refresh\_token\_ttl = **03:00 SAT**

# Validade do Token de Refresh (em segundos ou um determinado dia da semana com hh:MM)

# Dias da semana: SUN, MON, TUE, WED, THU, FRI e SAT

acceptor.domain.streamerfeedapi.streamers.keepsessionalive = **true**

# Streamer fará que o 'HeartBeat' seja um comando de atividade humana

acceptor.domain.streamerfeedapi.streamers.transport = **NoneAuto**

# (default: PersistentChannel) Transporte a ser utilizado pela sessão. Pode ser declarado como campo 'transport' no LoginRequest (não é campo obrigatório)

# 'PersistentChannel': Transporte pelo canal persistente PUSH AJAX/Comet (padrão)

# 'Polling': Canal persistente modo polling. Ao invés do servidor notificar as pontas via PUSH, as pontas fazem polling para obter os dados do servidor.

# 'None': Sem canal persistente. As pontas obtem dados através dos comandos quando 'sync=true', ou seja, através de snapshots.

# 'NoneAuto': Sem canal persistente. As pontas obtem dados através dos comandos quando 'sync=true', ou seja, através de snapshots.

Observações:

- Se utilizado assinatura a configuração deverá ser “*Polling*”;

- A configuração de “*transport*” também pode ser adicionada ao instanciar a API.

▪ Streamer:

session\_default\_transport = **NoneAuto**

# (default: PersistentChannel) Transporte a ser utilizado pela sessão. Pode ser declarado como campo 'transport' no LoginRequest (não é campo obrigatório)

# 'PersistentChannel': Transporte pelo canal persistente PUSH AJAX/Comet (padrão)

# 'Polling': Canal persistente modo polling. Ao invés do servidor notificar as pontas via PUSH, as pontas fazem polling para obter os dados do servidor.

# 'None': Sem canal persistente. As pontas obtem dados através dos comandos quando 'sync=true', ou seja, através de snapshots.

# 'NoneAuto': Sem canal persistente. As pontas obtem dados através dos comandos quando 'sync=true', ou seja, através de snapshots.

Observações:

- Se utilizado assinatura a configuração deverá ser “*Polling*”;

- A configuração de “*transport*” também pode ser adiciona na API ou Acceptor.

adv\_unsign\_quotes\_after\_inactivity\_of = **8h**

# (default: 60s) Tempo de inatividade de clientes antes de desassinar uma cotação no Advanced

idle\_session\_timeout = **5m**

# (default: 30m - minutos)- tempo limite de uma sessão válida, porém sem atividade humana

# 2.1.7) Configurações do Acceptor para utilizar “*login*” com “*token*”

▪ Configurações do Acceptor:

acceptor.domain.streamerfeedapi.accept\_login\_token = true

# Habilitar login com token

acceptor.domain.streamerfeedapi.logintoken.secret\_key = f1iE04/0JcYT+V1gWqG18A==

# AES Secret-Key (criptografado com Client-Crypto ⇨ client-crypto-1.0.0.jar)

acceptor.domain.streamerfeedapi.logintoken.iv\_key = 94P9TiXUtjWLhDPuKdBrkkxfyDqrRsfO

# ES IV-Key (criptografado com Client-Crypto ⇨ client-crypto-1.0.0.jar ⇨ Para API utilizar o valor ‘*IV\_VALUE\_16\_BYTE*’)

acceptor.domain.streamerfeedapi.logintoken.salt\_key = W7VvrFXd3TgVzzEwWq8j4g==

# AES Salt-Key (criptografado com Client-Crypto ⇨ client-crypto-1.0.0.jar)

acceptor.domain.streamerfeedapi.logintoken.salt\_iterations = 65536

# AES Salt Iterations

acceptor.domain.streamerfeedapi.logintoken.key\_len = 128

# Length of Key

acceptor.domain.streamerfeedapi.logintoken.expiration\_time = 120

# Tempo (em segundos) para expiração do token (se login\_token\_generator\_enabled=true)

acceptor.domain.streamerfeedapi.logintoken.ahead\_time = 120

# Tempo (em segundos) para controle de tempo à frente do token (se login\_token\_generator\_enabled=true)

acceptor.domain.streamerfeedapi.login\_token\_generator\_enabled = true

# Habilita (se true) o controle de expiração e tempo à frente (não permite utilizar um token já expirado ou com tempo à frente)

Atenção: O parâmetro “*logintoken.iv\_key*” deve ter o valor “*IV\_VALUE\_16\_BYTE*” e os demais campos devem ser customizados, principalmente “*logintoken.secret\_key*” e “*logintoken.salt\_key*” (ambos com no máximo 16 caracteres alfanuméricos).

▪ Instanciando a API:

apiParameters.LoginTokenSecretKey = "PASSWORD\_VALUE";

apiParameters.LoginTokenIvKey = "IV\_VALUE\_16\_BYTE";

apiParameters.LoginTokenSaltKey = "SALT\_VALUE";

apiParameters.LoginTokenSaltIterations = "65536";

apiParameters.LoginTokenKeyLength = "128";

⇨ Criptografar os valores utilizando Client-Crypto (client-crypto-1.0.0.jar):

LoginTokenSecretKey = PASSWORD\_VALUE = f1iE04/0JcYT+V1gWqG18A==;

LoginTokenIvKey = IV\_VALUE\_16\_BYTE = 94P9TiXUtjWLhDPuKdBrkkxfyDqrRsfO;

LoginTokenSaltKey = SALT\_VALUE = W7VvrFXd3TgVzzEwWq8j4g==;

⇨ Configurar Acceptor com as informações criptografadas no item anterior:

acceptor.domain.streamerfeedapi.logintoken.secret\_key = f1iE04/0JcYT+V1gWqG18A==

acceptor.domain.streamerfeedapi.logintoken.iv\_key = 94P9TiXUtjWLhDPuKdBrkkxfyDqrRsfO

acceptor.domain.streamerfeedapi.logintoken.salt\_key = W7VvrFXd3TgVzzEwWq8j4g==

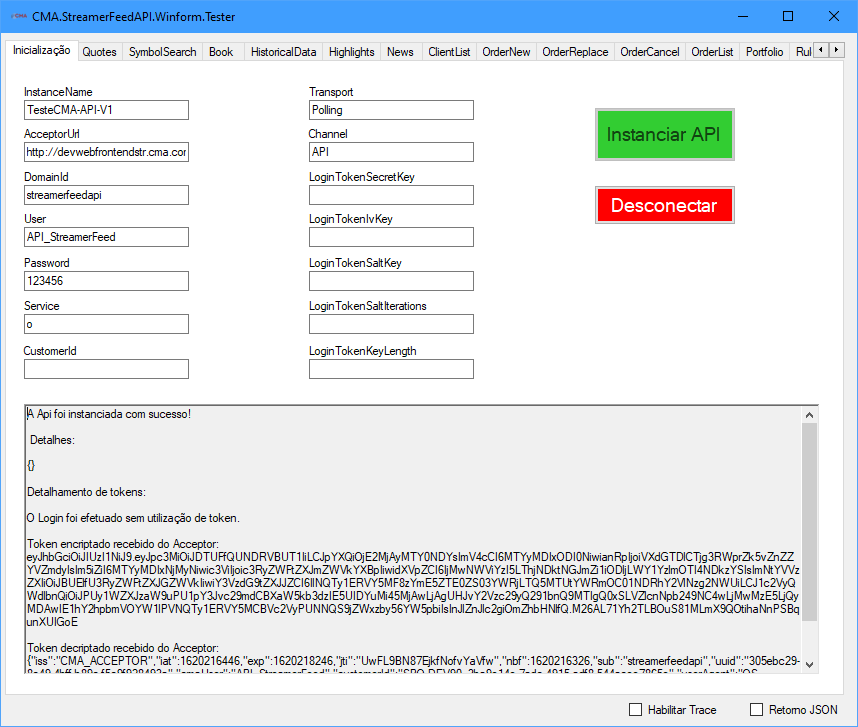
acceptor.domain.streamerfeedapi.logintoken.salt\_iterations = 65536

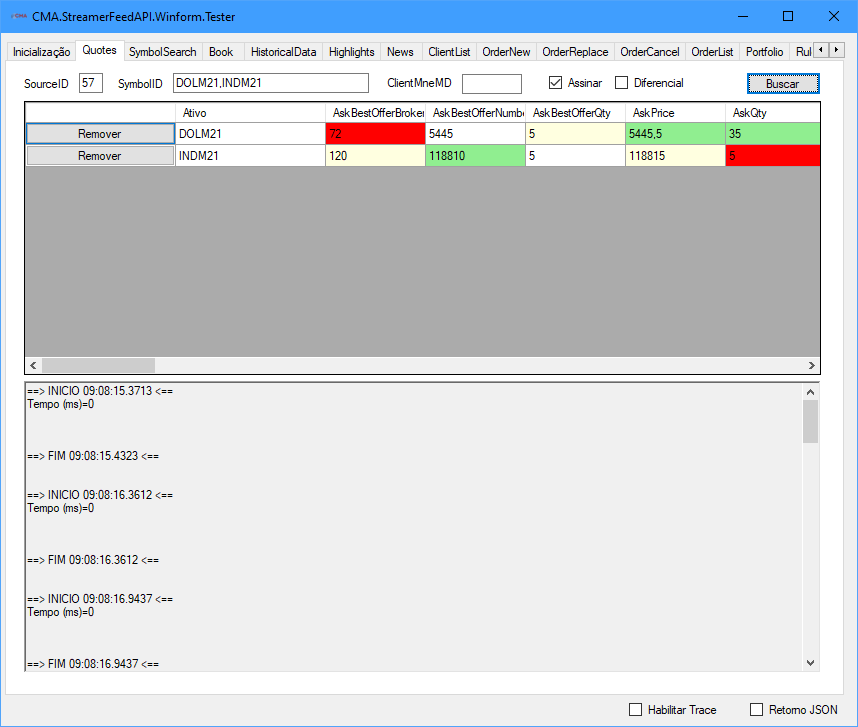
acceptor.domain.streamerfeedapi.logintoken.key\_len = 128

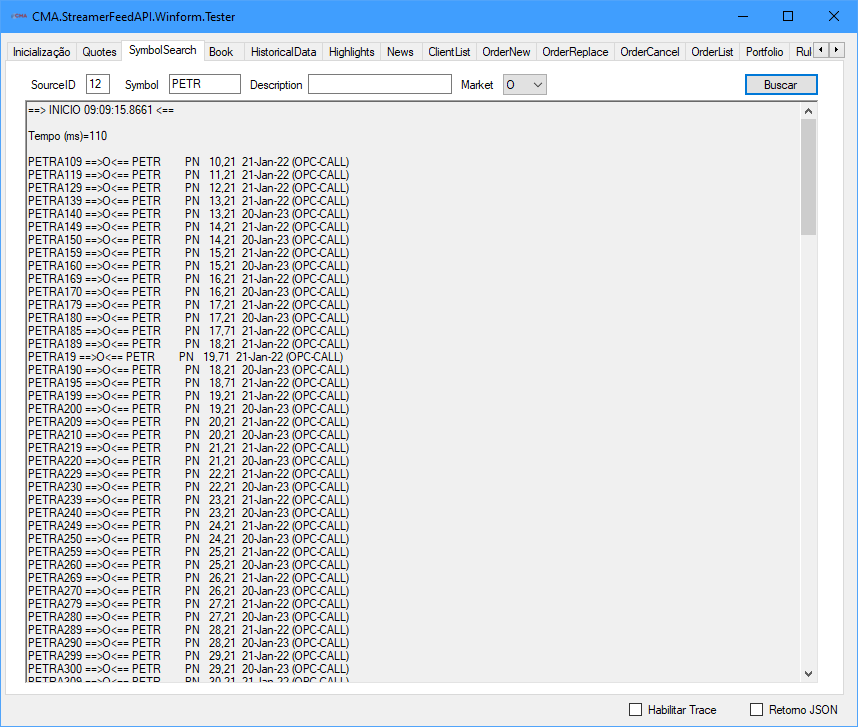
Observação: Funcionalidade adicionada a partir da versão 1.0.31 do Acceptor e 1.0.0.323 da API.

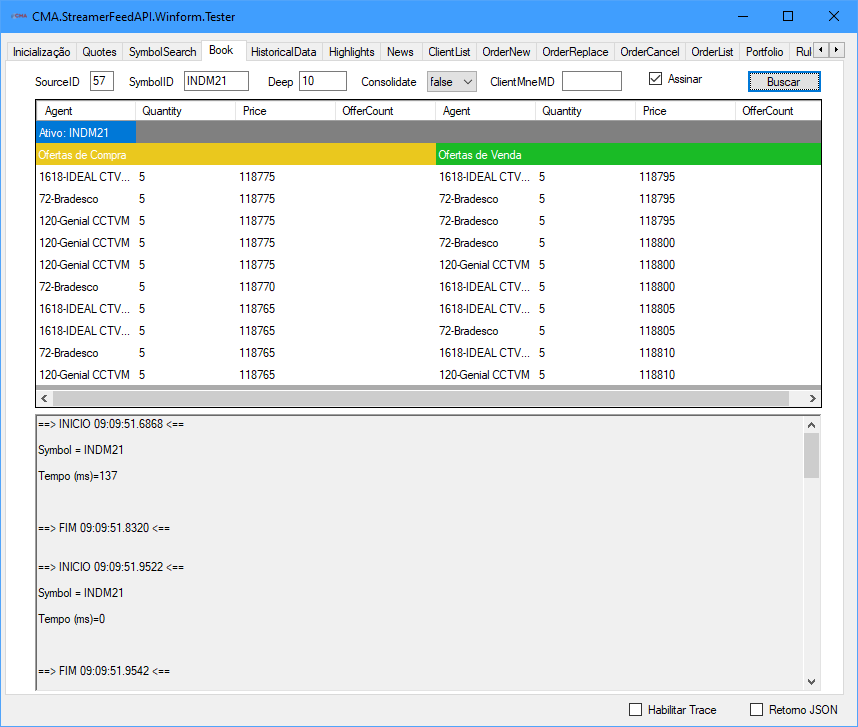
# 2.1.8) Exemplo com fonte utilizando CMA Streamer Feed API

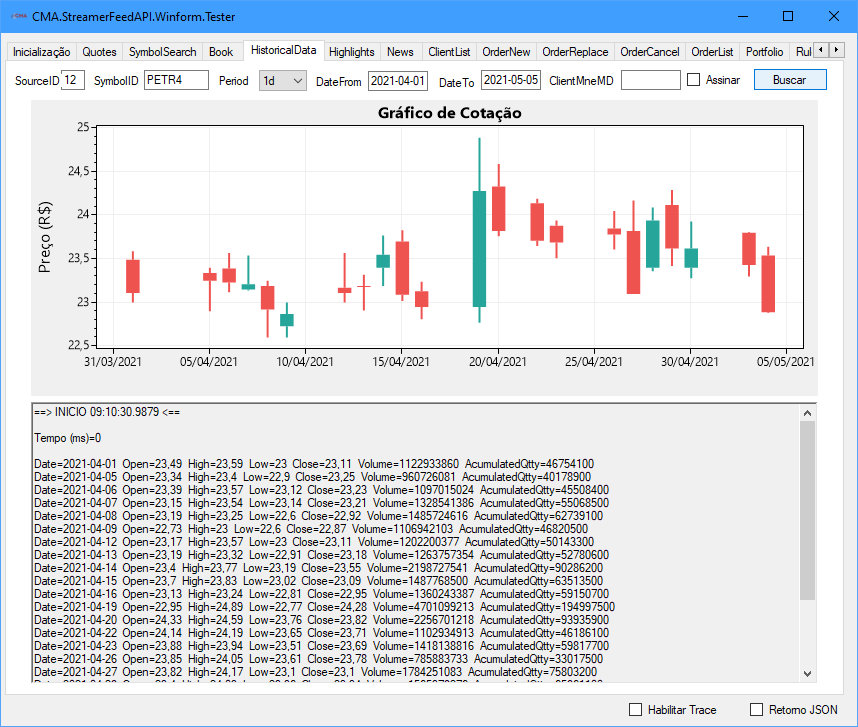
▪ Segue algumas telas de exemplo do CMA Streamer Feed API WinForm Tester:

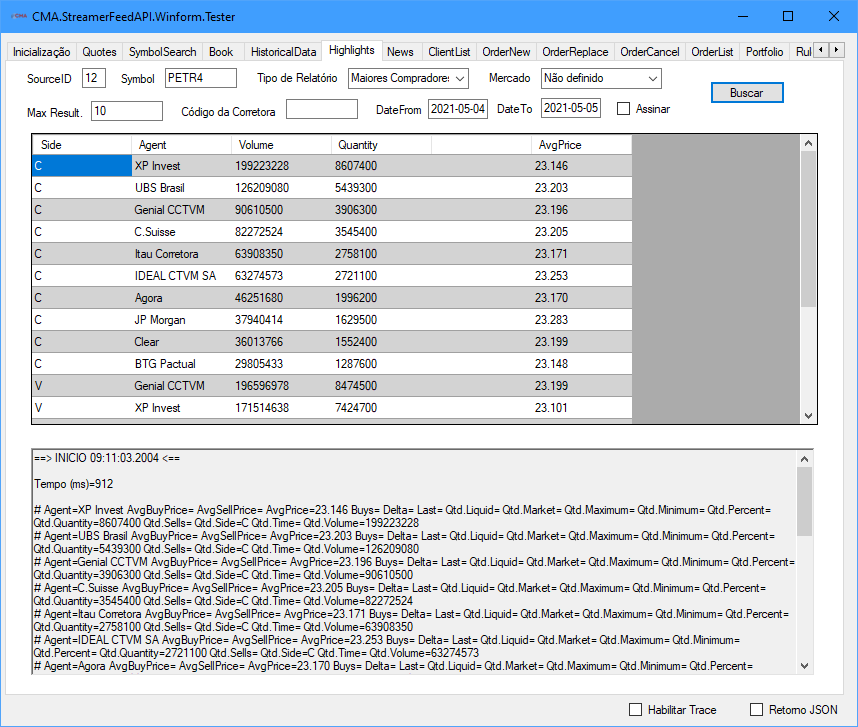


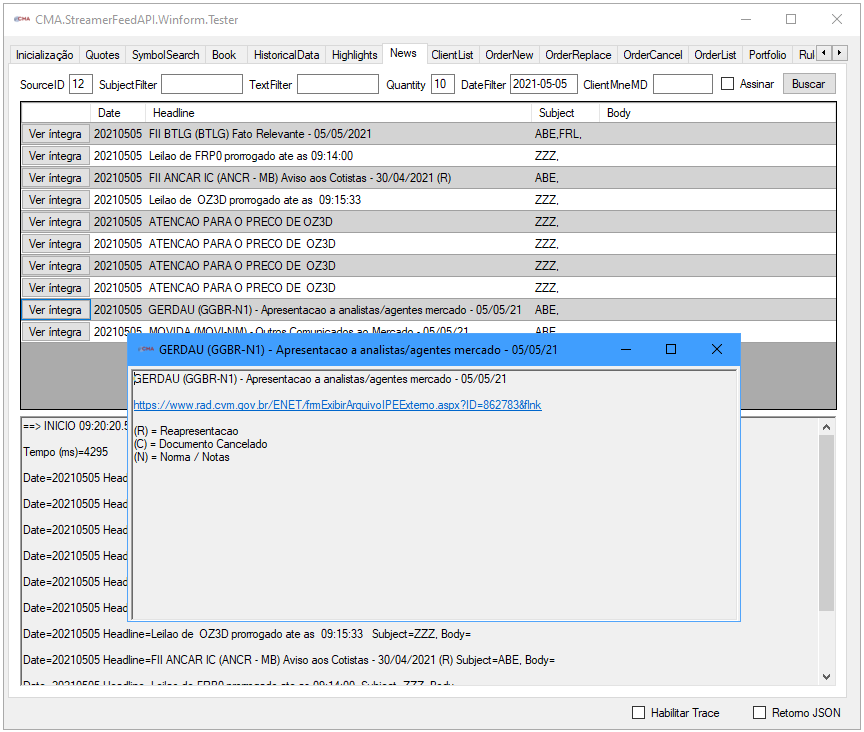


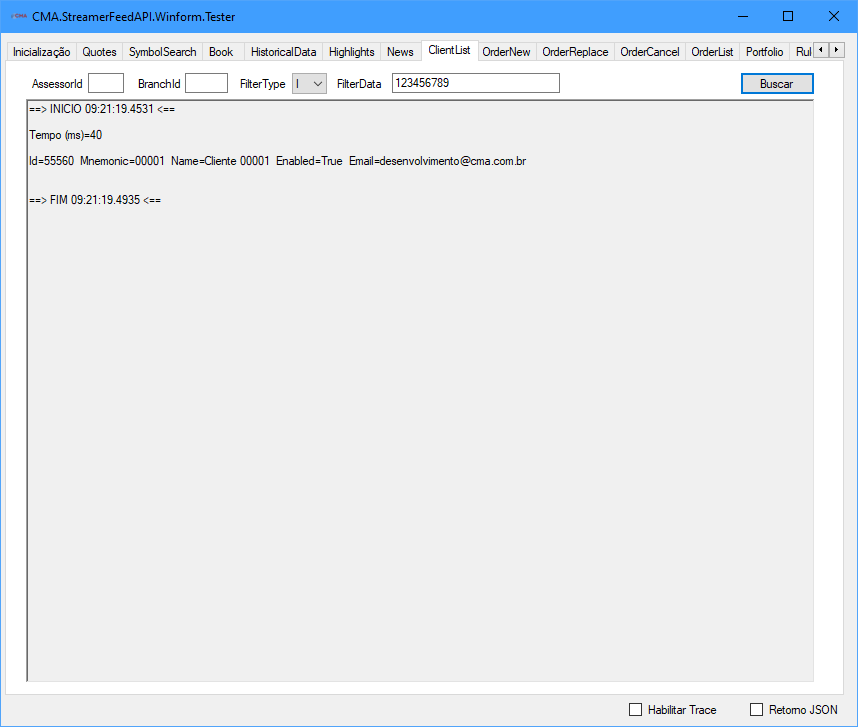


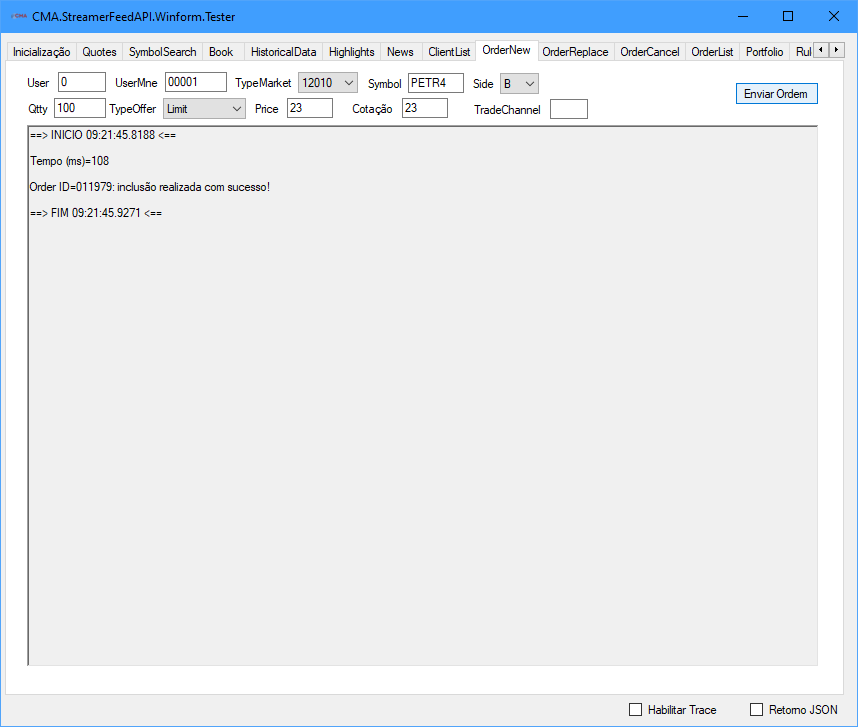


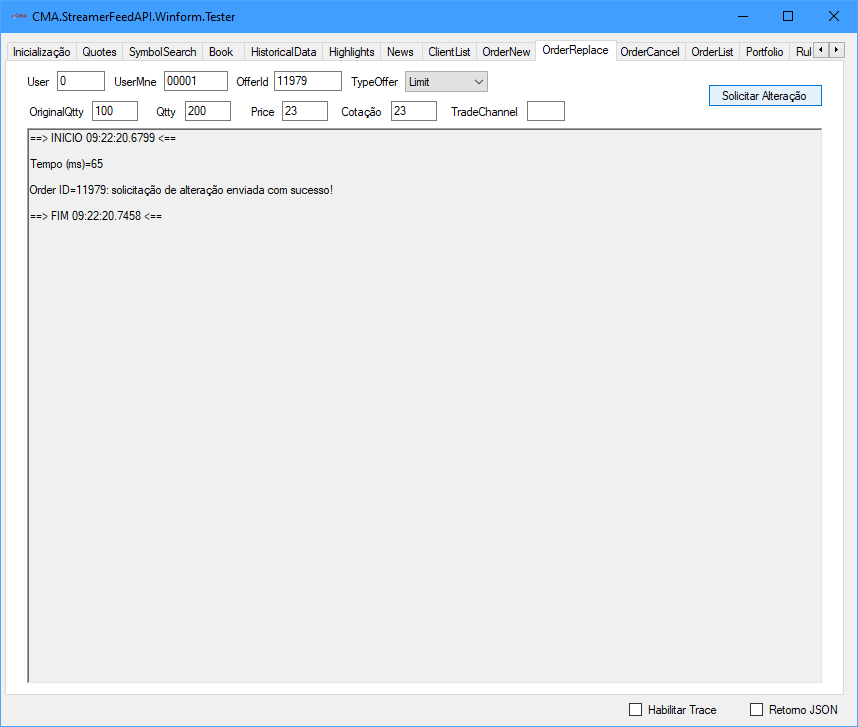


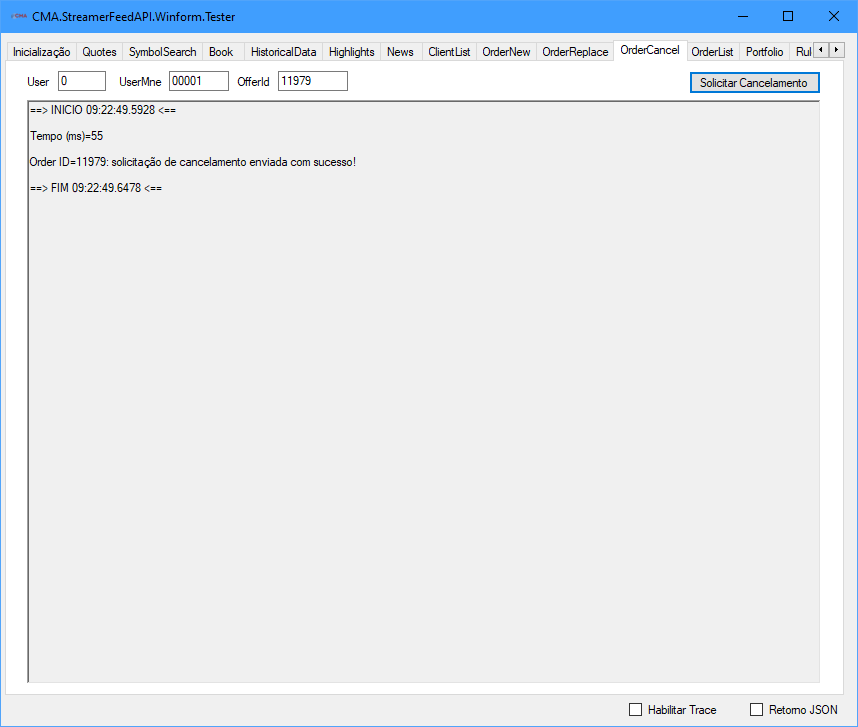


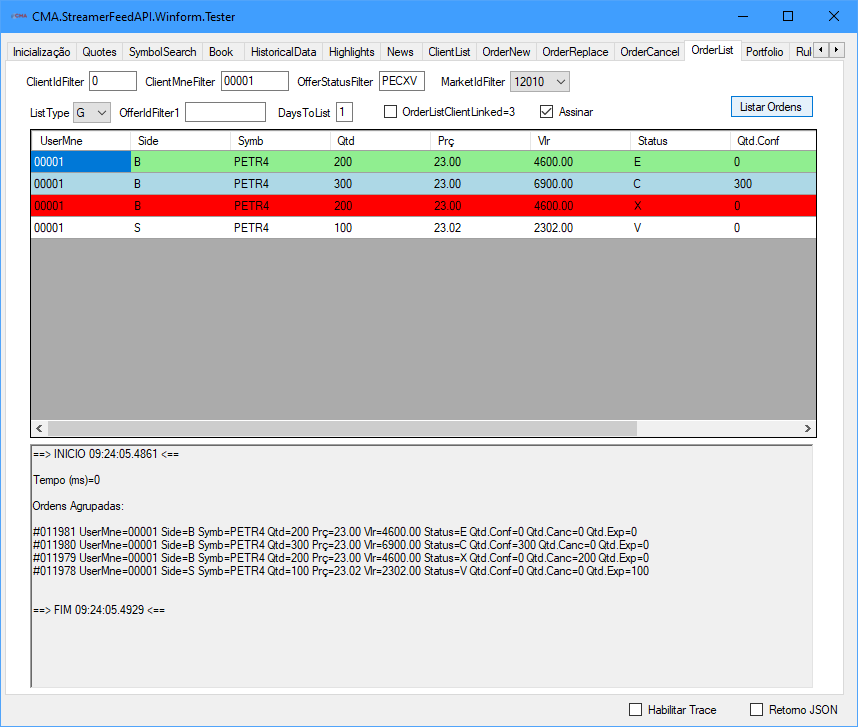


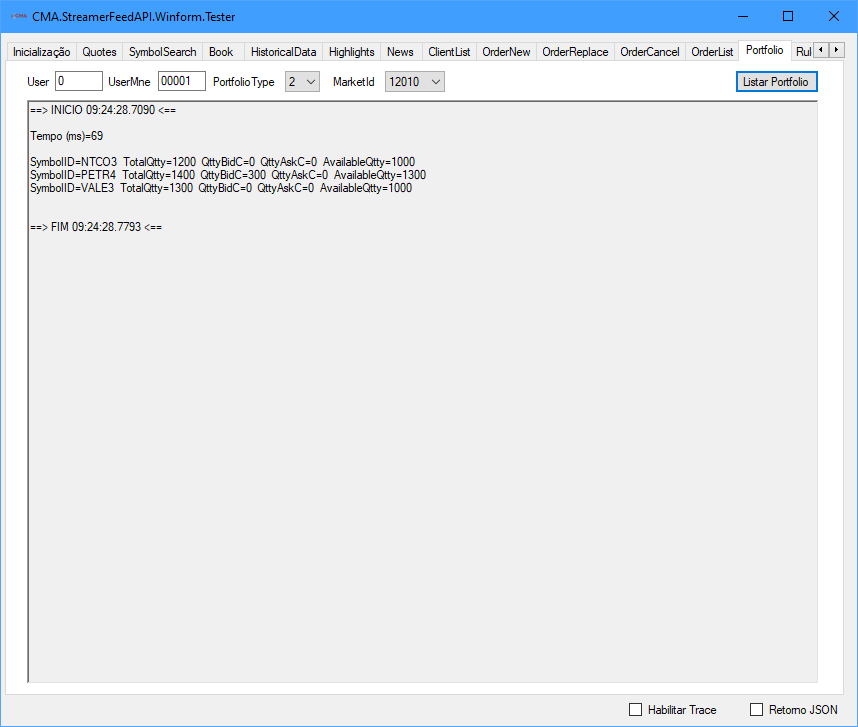


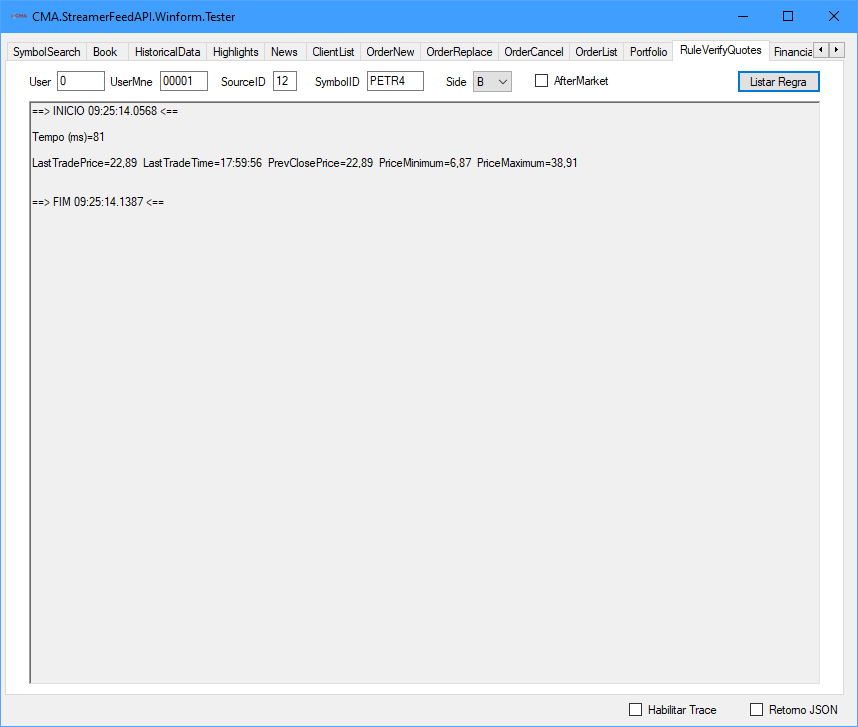


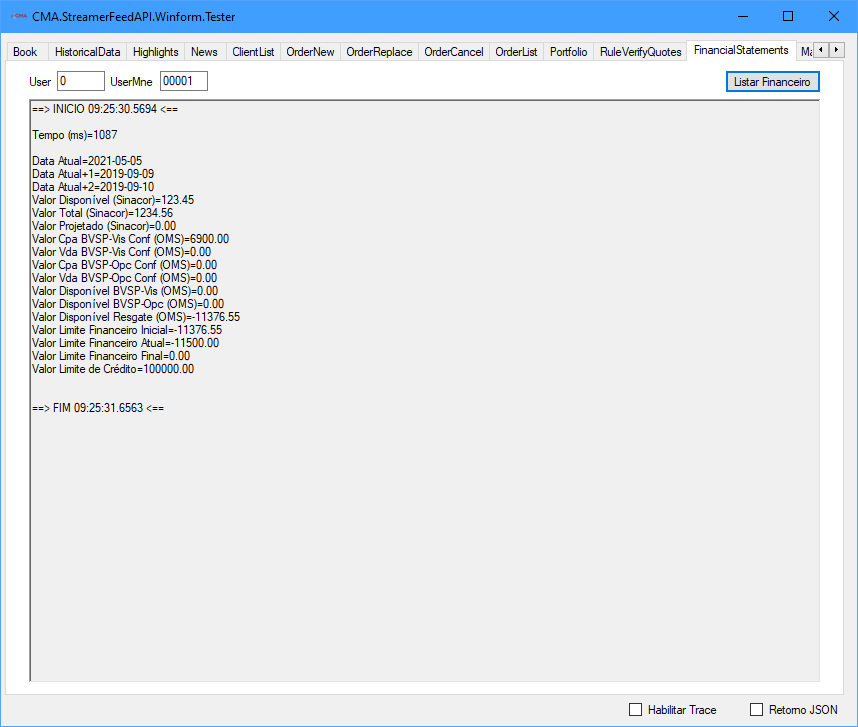


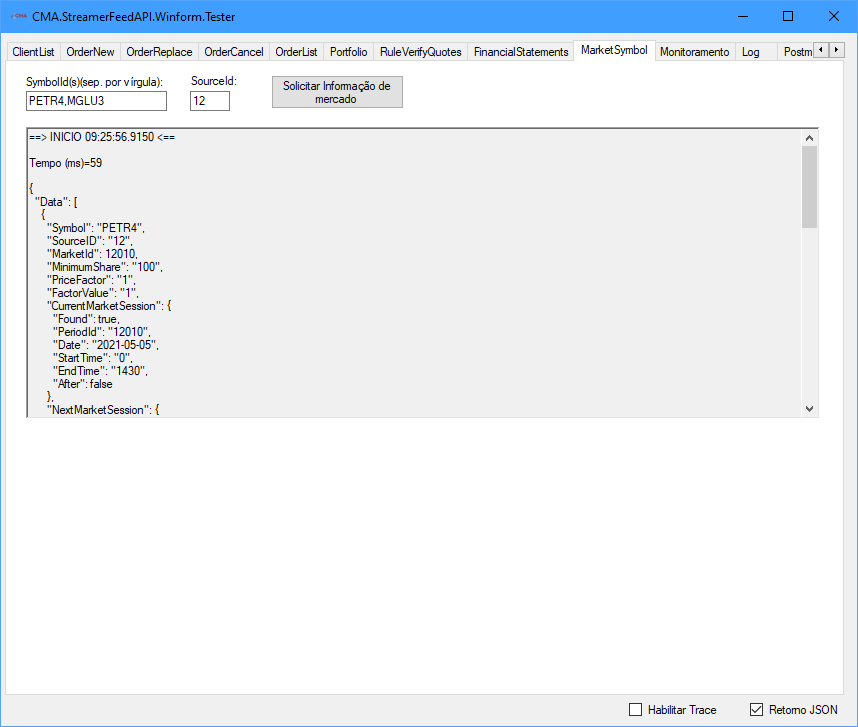


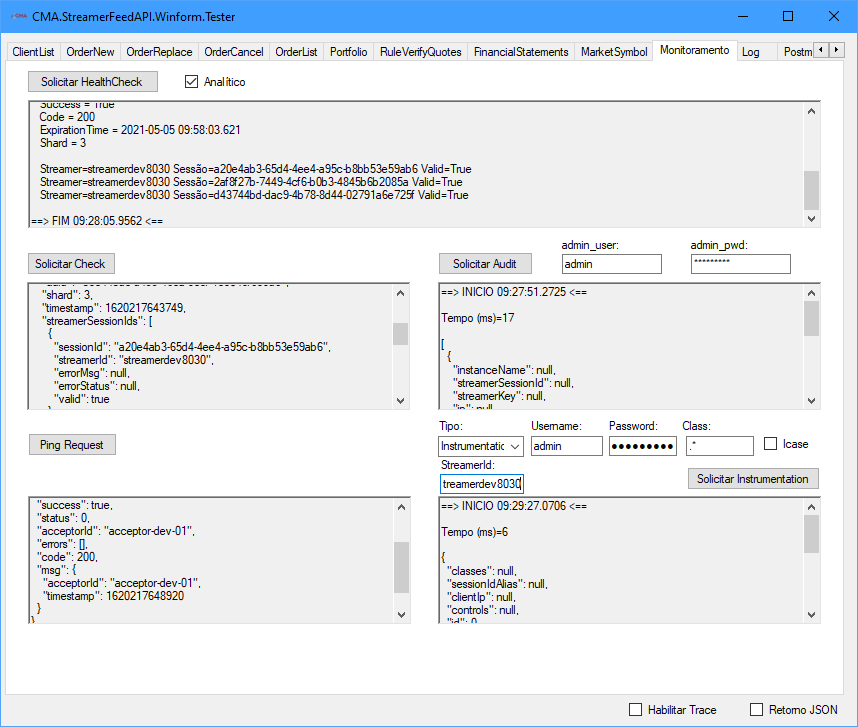


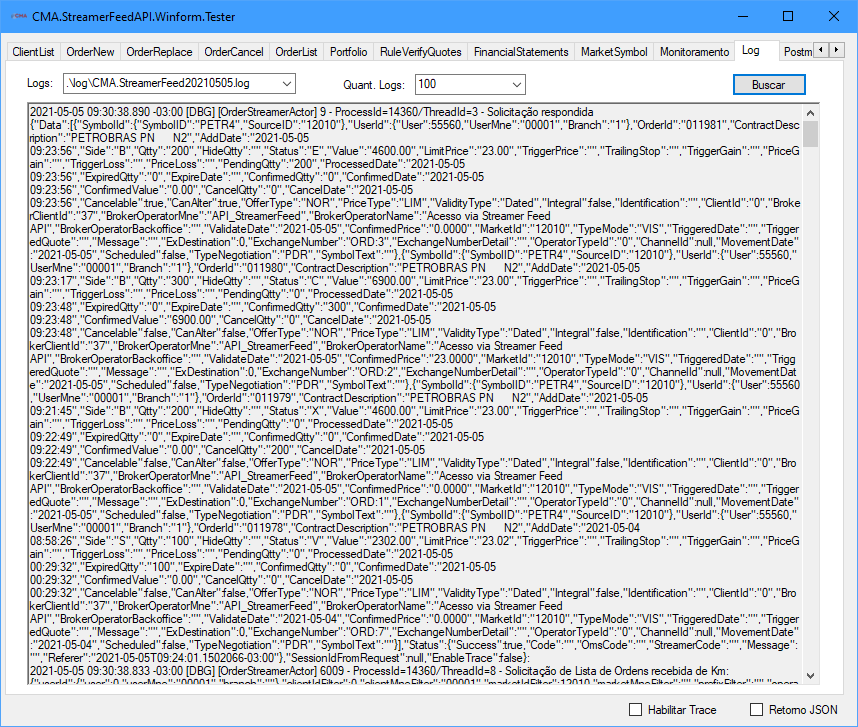












▪ Fontes para Microsoft .NET Framework 4.8:

- Disponibilizado juntamente com o pacote da API/REST (consulte Atendimento CMA).

# 2.2) Tratamento de Erros e Exceptions da API

▪ Uma “*exception*” será lançada em caso de erro na inicialização e nas respectivas requisições;

▪ No “*Result*” será informado se a requisição aconteceu com sucesso através do campo “*Status.Success=true*”;

▪ Se “*Status.Success=false*” as informações do erro serão disponibilizadas nos campos:

⇨ Status.Code (string) = Código do erro com a numeração:

- 00000 a 00999 = System;

- 01000 a 01999 = Modules;

- 02000 a 02999 = Conector;

- 04000 a 04999 = Api;

- 05000 a 05999 = MarketData;

- 06000 a 06999 = Trading;

- 07000 a 07999 = Commons;

- 08000 a 08999 = Monitor.

⇨ Status.Message (string) = Mensagem textual do erro;

⇨ Status.OmsCode (string) = Código do erro OMS quando retornado por este sistema;

⇨ Status.StreamerCode (string) = Código do erro Streamer.



▪ Lista de códigos de erros retornados pelo OMS no campo “*Status.OmsCode*”:



⇨ Regras do OMS (tipos Verificação, Limite Financeiro, Posição Custódia e Risco):

- ASSESSOR\_LIMIT\_1 - Considera os movimentos do dia e o limite financeiro cadastrado. Esta regra é aplicada nas operações de compra;

- ASSESSOR\_LIMIT\_2 - Considera os movimentos do dia e o limite financeiro cadastrado. É aplicada também em operações de Venda;

- ASSESSOR\_LIMIT\_3 - Considera apenas o limite financeiro. Esta regra é aplicada nas operações de compra;

- ASSESSOR\_LIMIT\_4 - Considera apenas o parâmetro de entrada como sendo o limite financeiro. Esta regra é aplicada nas operações de compra;

- ASSESSOR\_LIMIT\_5 - Considera o limite financeiro cadastrado e todas as negociações do dia (somando Compra e Venda, estilo MegaBolsa);

- BLOCK\_ILIQUID\_SYMBOL - Bloqueio de operação com símbolos sem liquidez;

- BLOCK\_SYMBOL - Regra que permite o bloqueio de ofertas para um determinado Mercado e Símbolo;

- BMF\_MANAGER - Repassador de Ordens BM&F;

- BMF\_OPERATOR - Operador Mesa BM&F;

- BVSP\_MANAGER - Gestor de Ordens BVSP;

- BVSP\_OPERATOR\_800 - Operador 800 BVSP;

- BVSP\_TRANSMITTER - Repassador de Ordens BVSP;

- CHECK\_QUOTATION\_1 - Calcula a faixa de valores permitidos para o preço limite;

- CHECK\_QUOTATION\_2 - Calcula a faixa de valores permitidos para o preço limite (no Período After-Market);

- CHECK\_QUOTATION\_3 - Calcula, para um determinado Mercado e Símbolo, a faixa de valores permitidos para o preço limite;

- CHECK\_QUOTATION\_4 - Calcula, para um determinado Mercado e Símbolo, a faixa de valores permitidos para o preço limite (no Período After-Market);

- CHECK\_TPMODE - Verifica Modalidade com quantidade solicitada (a quantidade precisa pertencer a modalidade);

- CLIENT\_LIMIT\_1 - Considera os movimentos do dia, limites financeiros cadastrados, limites do backoffice e saldo do backoffice. Permite também a entrada de um limite adicional;

- CLIENT\_LIMIT\_2 - Considera os movimentos do dia, os limites financeiros cadastrados, os limites no backoffice, o saldo no backoffice e o valor de alavancagem do cliente;

- CLIENT\_LIMIT\_3 - Considera os movimentos do dia, os limites financeiros cadastrados e os limites no backoffice. Esta regra é aplicada somente nas operações de compra;

- CLIENT\_LIMIT\_4 - Considera os movimentos do dia, os limites financeiros cadastrados e os limites no backoffice. Permite também a entrada de um limite adicional;

- CLIENT\_LIMIT\_5 - Limite diário de negociação (somando Compra e Venda, estilo MegaBolsa) para o mercado normal. Esta regra é aplicada nas operações de compra e venda;

- CLIENT\_LIMIT\_6 - Limite diário de negociação (somando Compra e Venda, estilo MegaBolsa) para o mercado after-market. Esta regra é aplicada nas operações de compra e venda;

- CLIENT\_LIMIT\_7 - Procedure customizada;

- CLIENT\_LIMIT\_8 - Limite do Cliente 8 (aplicada na BVSP compra e venda com possibilidade de venda a descoberto);

- CLIENT\_POSITION\_1 - Verifica posição em custódia do cliente (liberando compras d-1, d-2 e pendentes);

- CLIENT\_POSITION\_2 - Verifica posição em custódia do cliente (não liberando compras d-1,d-2 e pendentes);

- CLIENT\_POSITION\_3 - Verifica posição em custódia do cliente - bloqueio no ativo objeto;

- CLIENT\_POSITION\_4 - Verifica posição em custódia do cliente - Negocia quantidade pendente;

- COLLATERAL\_LIMIT - Limite garantia;

- FIXED\_LIMIT\_1 - Valida Estoque;

- FIXED\_POSITION\_1 - Posição Renda Fixa;

- FIX\_COM\_SUITABILITY - Suitability Renda Fixa;

- FOLLOW\_CLIENT\_CHECK - Segue Verificação do Cliente;

- FOLLOW\_CLIENT\_LIMIT - Segue Limite Financeiro do Cliente;

- FOLLOW\_CLIENT\_POSITI - Segue Posição em Custódia do Cliente;

- FOLLOW\_CLIENT\_RISK - Segue Risco do Cliente;

- MIN\_MAX\_1 - Determina valores mínimo e máximo para Compra e Venda;

- MIN\_MAX\_2 - Determina valores mínimo e máximo para Compra ou Venda para um determinado Símbolo (lembrando que podemos usar o filtro de mercado);

- ONLINE\_LIMIT\_PLUS - Regra Online+;

- ONLINE\_POSITION\_PLUS - Regra Venda Online+;

- OPERATOR\_LIMIT\_1 - Considera o limite financeiro cadastrado e todas negociações do dia (somando Compra e Venda, estilo MegaBolsa);

- OPERATOR\_LIMIT\_2 - Considera os movimentos do dia e o limite financeiro cadastrado. Esta regra é aplicada nas operações de compra;

- POS\_EXE\_FREE - Regra que permite negociar um ativo no dia de seu exercício/vencimento;

- POS\_EXE\_REVERSION - Regra que permite apensar reverter a posição do cliente até um determinado horário após o início do pregão;

- RISK\_CLIENT1 - Alav.Risco;

- RULE\_CHECK\_CATEGORY - Regra que permite redirecionar uma ordem para uma porta configurada de acordo com o tipo de cliente;

- RULE\_CHECK\_CHANNEL - Redirecionamento por canal BVSP;

- RULE\_CHECK\_CHN\_BVMF - Redirecionamento por canal BVMF;

- RULE\_CHECK\_PREFIX - Verifica Parâmetros do Prefixo (configurado na pasta de Direitos do Cliente ou Grupo de Cliente);

- RULE\_CHECK\_ZEXZE - Regra que permite verificar se o cliente está fazendo uma oferta que pode gerar negócio com ele mesmo;

- RULE\_COM\_1 - Chamada de Procedimento;

- RULE\_COM\_2 - Chamada de Procedimento;

- RULE\_COM\_SUITABILITY - Chamada de Procedimento Suitability.

⇨ Regras do OMS vs códigos de erros:

- Bloqueia Símbolo (BLOCK\_SYMBOL)

. RULE\_CHECK\_BLOCK\_SYMBOL: Market: %1%, Symbol: %2%. Price: %3%. Cannot be accepted!

- Verifica Cotação 1 (CHECK\_QUOTATION\_1)

- Verifica Cotação 2 (CHECK\_QUOTATION\_2)

- Verifica Cotação 3 (CHECK\_QUOTATION\_3)

- Verifica Cotação 4 (CHECK\_QUOTATION\_4)

. RULE\_QUOTES\_NOT\_FOUND: Market: %1%, Symbol: %2%. Quotes not found!

. RULE\_CHECK\_QUOTATION: Minimum Price: %1%, Maximum Price: %2%. Can't accept %3% with %4% price

- Verifica Ordem Zé com Zé (RULE\_CHECK\_ZEXZE)

. RULE\_CHECK\_ZEXZE: Offer "Ze with Ze". Cannot be accepted!

- Verifica Modalidade (CHECK\_TPMODE)

. CHECK\_TPMODE: Offer with invalid quantity for this mode

- Verifica Parâmetros do Prefixo (RULE\_CHECK\_PREFIX)

. RULE\_PREFIX\_NOTFOUND: Prefixo %1% não habilitado. Não é possível aceitar oferta para o símbolo %2%. (onde: %1=prefixo - %2=símbolo);

. RULE\_PREFIX\_ERROR1: Qtde mínima para operar %1% e máxima %2% (qtde da oferta %3%). Não é possível aceitar oferta. (onde: %1=qtde mínima - %2=qtde máxima %3-qtde oferta);

. RULE\_PREFIX\_ERROR2: Qtde máxima em aberto (no(a) %1%) no %2% ultrapassada (qtde em aberto: %3% - qtde máxima: %4%) . Não é possível aceitar oferta. (onde: %1=texto ‘Dia’ ou ‘Posição’ - %2=texto ‘Prefixo’ ou ‘Símbolo’ - %3=valor PA - %4=qtde máxima).

- Regra COM (RULE\_COM\_1 e RULE\_COM\_2)

. RULE\_COM\_PROCEDURE: %1% (onde: %1=mensagem de retorno do componente - TAG\_ReturnMessage)

- Bloqueia Símbolo com baixa liquidez (BLOCK\_ILIQUID\_SYMBOL)

. RULE\_BLOCK\_ILIQUID\_SYMBOL: Símbolo ''%1%'' não possui liquidez, e esta bloqueado para negociação!

- Limite do Cliente 1 (CLIENT\_LIMIT\_1)

- Limite do Cliente 2 (CLIENT\_LIMIT\_2)

- Limite do Cliente 3 (CLIENT\_LIMIT\_3)

- Limite do Cliente 4 (CLIENT\_LIMIT\_4)

- Limite do Cliente 5 (CLIENT\_LIMIT\_5)

- Limite do Cliente 6 (CLIENT\_LIMIT\_6)

- Limite do Cliente 8 (CLIENT\_LIMIT\_8)

- Limite Online + (ONLINE\_LIMIT\_PLUS)

. RULE\_CLIENT\_LIMIT: Client %1%: Current Limit at %3%! Can't accept '%2%' Offer with %4% value!

- Limite do Cliente 7 (CLIENT\_LIMIT\_7)

. RULE\_PERSONAL\_LIMIT: Client %1%: Procedure Limit at %3%! Can't accept '%2%' Offer with %4% value!

- Limite do Assessor 1 (ASSESSOR\_LIMIT\_1)

- Limite do Assessor 2 (ASSESSOR\_LIMIT\_2)

- Limite do Assessor 3 (ASSESSOR\_LIMIT\_3)

- Limite do Assessor 4 (ASSESSOR\_LIMIT\_4)

- Limite do Assessor 5 (ASSESSOR\_LIMIT\_5)

. RULE\_ASSESSOR\_LIMIT: Assessor %1%: Current Limit at %3%! Can't accept '%2%' Offer with %4% value!

- Mínimo e Máximo 1 (MIN\_MAX\_1)

- Mínimo e Máximo 2 (MIN\_MAX\_2)

. RULE\_MAXMIN\_LIMIT: Minimum value allowed: %1%, Maximum value allowed: %2%. Can't accept '%3%' Offer with %4% value!

- Limite do Operador 1 (OPERATOR\_LIMIT\_1)

- Limite do Operador 2 (OPERATOR\_LIMIT\_2)

. RULE\_OPERATOR\_LIMIT: Operator %1%: Current Limit at %3%! Can't accept '%2%' Offer for Client %5% with %4% value!

- Custódia do Cliente 1 (CLIENT\_POSITION\_1)

- Custódia do Cliente 2 (CLIENT\_POSITION\_2)

. RULE\_POSITION\_CLIENT: Client %1%: Current Position at %3%!, Can't accept '%2%' Offer with %4% quantities!

- Negociação livre no exercício (POS\_EXE\_FREE)

. POS\_EXE\_FREE\_WARNING: Symbol %1% negociating on exercise date.

- Reversão de posição no exercício (POS\_EXE\_REVERSION)

. Se disparada pela quantidade: POS\_EXE\_REVERSION\_WARNING: Only position reversion allowed on symbol exercise date!

. Se disparada pelo horário: POS\_EXE\_REVERSION\_WARNING\_2: Position reversion allowed on symbol exercise date until %1%:%2%

- Custódia do Cliente 3 (CLIENT\_POSITION\_3)

. RULE\_POSITION\_CLIENT: Client %1%: Current Position at %3%!, Can't accept '%2%' Offer with %4% quantities!

. RULE\_POSITION\_CLIENT\_LEVERAGE (quando Limite Adicional > 0): Client %1%: Current Position at %3% and leverage at %5%!, Can't accept '%2%' Offer with %4% quantities!

- Posição Online + (ONLINE\_POSITION\_PLUS)

. RULE\_NO\_LIMIT\_CC: "Client %1% : No Limit!\r\nCan't accept '%2%' Offer with %4% value !",

. RULE\_CLIENT\_LIMIT: Client %1%: Current Limit at %3%! Can't accept '%2%' Offer with %4% value!

⇨ Exemplo JSON de retorno para erro de limite (1, 2, 3, 4, 5, 6 ou 8):

{"Success":false,"Code":"06013","OmsCode":"SVR5002","StreamerCode":"","Message":"# # SVRL5002 - 200000 # #\r\nCliente 00001 : limite atual em -1545.00 !\r\nNão é possível aceitar oferta <PETR4> com o valor 1545.00 !","Referer":null}

⇨ Exemplo JSON de retorno para erro de ação inexistente:

{"Success":false,"Code":"06013","OmsCode":"SVR0606","StreamerCode":"","Message":"# # SVR0606 # #\r\nNão pode ser encontrado Mercado/Símbolo <#12010>/<XXXX> na base de dados!\r\nOperação abortada!","Referer":null}

# 2.3) Instanciando a API

▪ A API deve ficar instanciada durante toda a “*vida*” funcional do serviço (durante o processo de instanciamento é realizado a conexão com os serviços CMA Acceptor, CMA Streamer e CMA OMS);

▪ A autenticação com o CMA OMS (Negociação + MarketData) acontece através do User/Password enviado no momento da criação da instância, esta autenticação acontece utilizando um “*operador*” do OMS, por isso é necessário cadastramento (via módulo AC) de um, ou mais operadores com regras determinadas pelo gestor da Corretora e com acesso restrito ao canal “*API*” (neste caso Service = “*o*”);

▪ Também é possível realizar autenticação no CMA Advanced (apenas MarketData) através de User/Password (neste caso Service = “*m*”);

▪ Parâmetros de entrada = Estrutura **InstanceParams**:

⇨ InstanceName (obrigatório): Nome da Instância (exemplo: TesterAPI-V1);

⇨ AcceptorUrl (obrigatório): URL do CMA Web Acceptor:

▫ Utilize endereço único para utilização via “*load balance*”;

▫ Ou lista de URL separadas por “;” (lembrando que estes Acceptors devem estar clusterizados).

⇨ DomainId (obrigatório): Domain de Streamers;

⇨ User e Password (obrigatório): Mnemônico e Senha do Operador OMS (onde os registros de transações serão alocados cadastrado no OMS via módulo AC) ou Login Advanced;

⇨ Service (obrigatório): Tipo de serviço utilizado “o” (Negociação + MarketData) ou “m” (apenas MarketData);

⇨ CustomerId (opcional): Identificação única do usuário da API (se não enviado a API vai utilizar “*hostname*” + GUID);

⇨ Transport (opcional): Configurar com “*Polling*” para canal persistente em modo Polling quando utilizado “*assinatura*”, ou “*NoneAuto*” sem canal persistente recebendo informações via Snapshot (se não enviado será utilizado o “*Transport*” configurado no Acceptor ou Streamer);

⇨ Channel (opcional): Identificação do canal a ser utilizado no login do OMS (se não informado será enviado o canal configurado no Acceptor);

⇨ LoginTokenSecretKey (opcional): AES Secret-Key (se informado indica que será utilizado Token);

⇨ LoginTokenIvKey (opcional): ES IV-Key (obrigatório se informado LoginTokenSecretKey);

⇨ LoginTokenSaltKey (opcional): AES Salt-Key (obrigatório se informado LoginTokenSecretKey);

⇨ LoginTokenSaltIterations (opcional): AES Salt Iterations (obrigatório se informado LoginTokenSecretKey);

⇨ LoginTokenKeyLength (opcional): Length of Key (obrigatório se informado LoginTokenSecretKey).

Observação: Login por Token foi adicionado a partir da versão 1.0.31 do Acceptor e 1.0.0.323 da API.

▪ Exemplo de como instanciar a API:

using System;

using CMA.StreamerFeed.Models.Interfaces;

...

try

{

// Criar estrutura para os parâmetros

var apiParameters = new CMA.StreamerFeed.Models.V1.InstanceParams();

apiParameters.InstanceName = "TesterAPI-V1";

apiParameters.AcceptorUrl = "http://127.0.0.1:9080/acceptor";

apiParameters.DomainId = "streamerfeedapi";

apiParameters.User = "desenv";

apiParameters.Password = "123456";

apiParameters.Service = "o";

apiParameters.CustomerId = "cma-service-001";

// Instanciar a API do Streamer Feed

IRuntimeInstance cmaStreamerFeed = CMA.StreamerFeed.Api.Runtime.Instantiate(apiParameters);

...

}

catch (Exception ex)

{

// Tratar exception

Console.WriteLine(ex.Message);

}

...

# 2.4) Solicitação de Informações de Market Data para API

▪ As informações de “*Market Data*” são recebidas através dos servidores CMA Streamer conectados com os servidores CMA Advanced (servidor de cotações, notícias, gráficos, páginas fixas, etc).

# 2.4.1) Solicitação de Cotações para API (Quotes)

▪ Método **Quotes**;

▪ Utilizar o método para uso ***Sem Assinatura***:

Quotes(IList<V1.Marketdata.Quotes.QuotesParam> quotesParams, int millisecondsTimeOut = 10000)

▪ Utilizar o método para uso ***Com Assinatura (online)***:

Quotes(IList<V1.Marketdata.Quotes.QuotesParam> quotesListParam, bool sign, CancellationToken cancelToken = default, int millisecondsTimeOut = 10000, bool onlyDiff = false)

Onde:

- sign = Utilizar true para assinatura;

- CancellationToken = Token que permite cancelar a assinatura (gerar token com o comando “*new CancellationTokenSource()*”);

- onlyDiff = Utilizar true para receber apenas os campos que sofreram alteração ou false para receber todos os campos na atualização.

*Atenção*: Utilizar o token gerado para efetuar o cancelamento da assinatura (comando “*cancellationTokenSource.Cancel();*”) que não está sendo mais utilizada, permitindo assim reduzir o consumo de memória, processador e rede (observação: se assinado vários símbolos todos serão cancelados).

▪ Parâmetros de entrada = Coleção da estrutura **QuotesParam**:

⇨ VendorId [opcional] (integer) = Id do Vendor:

⬝ 1 (default) = RealTime;

⬝ 3 = SnapShot (foto do momento);

⬝ 4 = SnapShot End-Day (foto do final do dia);

⬝ 7 = Contribuidor;

⬝ 8 = DelayTime.

Observação: Vendor solicitado deve estar liberado no Advanced.

⇨ SourceID [opcional] (string) = Id da Origem do símbolo (exemplo: “12” para Bovespa);

⇨ SymbolID [obrigatório] (string) = Símbolo (exemplo: PETR4);

⇨ Referer [opcional] (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ ClientMneMD (opcional) (string) = Mnemônico do Cliente OMS para contabilização de “*fee*” do Market Data;

⇨ EnableTrace (bool) = Vide item 2.3.1.

▪ Retorno = Coleção da estrutura **QuotesResult**:

⇨ Status = Status da cotação:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura QuotesResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Dados da cotação:

⬝ VendorId (integer) = Id do Vendor;

⬝ VendorDesc (string) = Descrição do Vendor;

⬝ Symbol:

- SourceId (string) = Bolsa de origem;

- SymbolID (string) = Código do símbolo (ticker da bolsa);

⬝ AskBestOfferBroker (int) = Id da corretora da melhor oferta de compra;

⬝ AskBestOfferNumber (double) = Número da melhor oferta de venda;

⬝ AskBestOfferQty (int) = Qtde da melhor oferta de venda;

⬝ AskPrice (double) = Preço de venda;

⬝ AskQty (int) = Qtde de venda;

⬝ AuctionAsk (bool) = Se true indica que o preço AskPrice é de leilão;

⬝ AuctionBid (bool) = Se true indica que o preço BidPrice é de leilão;

⬝ AvgPrice (double) = Preço médio;

⬝ BidBestOfferBroker (int) = Id da corretora da melhor oferta de compra;

⬝ BidBestOfferNumber (double) = Número da melhor oferta de compra;

⬝ BidBestOfferQty (int) = Qtde da melhor oferta de compra;

⬝ BidPrice (double) = Qtde de venda;

⬝ BidQty (int) = Qtde de compra;

⬝ CalendarDays (int) = Número de dias corridos para o vencimento;

⬝ Change (double) = Variação do último negócio (relação ao fechamento);

⬝ ClosePrice (double) = Preço fechamento;

⬝ CumulativeQuantity (long) = Qtde acumulada;

⬝ ExecDate (string) = Data de exercício no formato DD/MM/AAAA (exemplo: "05/06/2018");

⬝ ExecPrice (double) = Preço de exercício;

⬝ HighPrice (double) = Preço máximo;

⬝ LastDifferential (double) = Diferença entre o preço de fechamento e último.

⬝ LastTradeDate (string) = Data do último negócio no formato DD/MM/AA (exemplo: "05/06/18");

⬝ LastTradePrice (double) = Preço do último negócio;

⬝ LastTradeQty (int) = Qtde do último negócio;

⬝ LastTradeTime (string) = Horário do Último Preço no formato HH:MM:SS (exemplo: 11:53:22);

⬝ LowPrice (double) = Preço mínimo;

⬝ MonthChange (string) = %Variação no mês (exemplo: "+7.33");

⬝ MonthsChange (string) = %Variação 30 dias (exemplo: "+48.67");

⬝ OpenContracts (int) = Número de contratos abertos.

⬝ OpenPrice (double) = Preço abertura;

⬝ PrevClosePrice (double) = Preço do fechamento anterior;

⬝ QuoteFactor (int) = Fator de cotação;

⬝ RemainingDaysToDue (int) = Número de dias para o vencimento;

⬝ Return (int) = Retorno;

⬝ Status (int) = Situação do símbolo com os códigos para Bovespa:

- 02 = Trading halt (Pause);

- 04 = No-Open (Close);

- 17 = Ready to trade (Open);

- 18 = Not available for trading (Forbidden);

- 20 = Unknown or invalid;

- 21 = Pre-Open (Reserved) 🡪 Leilão;

- 101 = Final Closing Call.

⬝ StockDetail (string) = Descrição do símbolo;

⬝ StockLength (int) = Fator de Negociação;

⬝ Tendency (string) = Tendência (+, - ou =) do último preço em relação ao anterior;

⬝ TradesCount (int) = Número de negócios;

⬝ Volume (double) = Volume;

⬝ Workday (int) = Número de dias úteis para o vencimento;

⬝ YearChange (string) = %Variação 12 meses (exemplo: "+14.81").

Observação: Os campos podem conter “null” informando que não existe cotação para o campo (exemplo: antes da abertura do mercado não temos LastTradePrice).

▪ 1º Exemplo solicitando cotações da PETR4 assincronamente (timeout de 10 segundos):

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Marketdata.Quotes;

...

// Lista de QuotesParam

IList<QuotesParam> quotesParamList = new List<QuotesParam>();

// PETR4

quotesParam = new QuotesParam();

quotesParam.SourceID = "12";

quotesParam.SymbolID = "PETR4";

quotesParamList.Add(quotesParam);

// Solicitar cotação

cmaStreamerFeed.Quotes(quotesParamList, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(IList<Result<QuotesResult>> results)

{

foreach (Result<QuotesResult> result in results)

{

if (result.Status.Success)

{

Console.WriteLine($"\nCotação do símbolo {result.Data.Symbol.Symbol} é:" + result.Data.LastTradePrice);

}

else

{

Console.WriteLine($"\nSímbolo {result.Data.Symbol.Symbol} erro-code: {result.Status.Code}");

}

}

}

▪ 2º Exemplo solicitando uma lista de cotações da PETR4 e ITUB4 assincronamente com retorno em JSON (timeout de 10 segundos):

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Marketdata.Quotes;

...

// Lista de QuotesParam

IList<QuotesParam> quotesParamList = new List<QuotesParam>();

// PETR4

QuotesParam quotesParam1 = new QuotesParam();

quotesParam1.SourceID = "12";

quotesParam1.SymbolID = "PETR4";

quotesParamList.Add(quotesParam1);

// ITUB4

QuotesParam quotesParam2 = new QuotesParam();

quotesParam2.SourceID = "12";

quotesParam2.SymbolID = "ITUB4";

quotesParamList.Add(quotesParam2);

// Solicitar cotações

cmaStreamerFeed.Quotes(quotesParamList, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriterJSON(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriterJSON(IList<Result<QuotesResult>> results)

{

foreach (Result<QuotesResult> result in results)

{

if (result.Status.Success)

{

Console.WriteLine($"\nCotação do símbolo {result.Data.Symbol.SymbolID} é:" + result.Data.ToJSON());

}

else

{

Console.WriteLine($"\nSímbolo {result.Data.Symbol.SymbolID} erro-code: {result.Status.Code}");

}

}

}

▪ 3º Exemplo solicitando cotações da PETR4 sincronamente (timeout de 10 segundos):

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Marketdata.Quotes;

...

// Lista de QuotesParam

IList<QuotesParam> quotesParamList = new List<QuotesParam>();

// PETR4

QuotesParam = new QuotesParam();

quotesParam.SourceID = "12";

quotesParam.SymbolID = "PETR4";

quotesParamList.Add(quotesParam);

// Solicitar cotação

cmaStreamerFeed.Quotes(quotesParamList, 10000).Subscribe(

obs =>

{

try

{

SyncResultTestWriter(obs.Result);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void SyncResultTestWriter (IList<Result<QuotesResult>> results)

{

foreach (Result<QuotesResult> result in results)

{

if (result.Status.Success)

{

Console.WriteLine($"\nCotação do símbolo {result.Data.Symbol.Symbol} é:" + result.Data.LastTradePrice);

}

else

{

Console.WriteLine($"\nSímbolo {result.Data.Symbol.Symbol} erro-code: {result.Status.Code}");

}

}

}

# 2.4.2) Solicitação de Lista de Ações/Opções para API (SymbolSearch)

▪ Método **SymbolSearch**;

▪ Parâmetros de entrada = Estrutura **SymbolSearchParam**:

⇨ SourceID [opcional] (string) = Id da Origem do símbolo (exemplo: 12 – Bovespa);

⇨ Symbol [opcional] (string) = Código da ação/opção/contrato/etc a ser pesquisada;

⇨ Description [opcinal] (string) = Descrição da ação/opção/contrato/etc a ser pesquisada;

⇨ Market [opcinal] (string) = Mercado do símbolo a ser pesquisado:

- E = Exercício de opções;

- F = Ação fracionário;

- G = Geral;

- I = Índice;

- J = Índice opções;

- O = Opção;

- R = Outros;

- T = Termo;

- U = Futuro;

- V = Ação vista;

- X = Não definido.

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

▪ Retorno = Estrutura **SymbolSearchResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura SymbolSearchResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com dados da lista de símbolos (pode retornar sem nenhum símbolo):

⬝ SourceID (string) = Código da origem;

⬝ Symbol (string) = Busca pelo código do símbolo (ticker da bolsa);

⬝ Description (string) = Busca pela descrição do símbolo;

⬝ Market (string) = Mercado do símbolo:

- E = Exercício de opções;

- F = Ação fracionário;

- G = Geral;

- I = Índice;

- J = Índice de opções;

- O = Opção;

- R = Outros;

- T = Termo;

- U = Futuro;

- V = Ação vista;

- X = Não definido.

⬝ Enabled (bool) = True ou False;

⬝ Indexes (List<string>) = Lista de índices que este símbolo está inserido (exemplo: IBOV);

⬝ Currency (string) = Moeda do símbolo;

⬝ TimeOpen (string) = Horário inicial pregão normal (formato HH:MM);

⬝ TimeClose (string) = Horário final pregão normal (formato HH:MM);

⬝ AfterOpen (string) = Horário inicial do after-market (formato HH:MM);

⬝ AfterClose (string) = Horário final do after-market (formato HH:MM);

⬝ NowExc (string) = Data/Hora da solicitação.

▪ Exemplo assíncrono para solicitar lista de símbolos dado um filtro:

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Marketdata.Symbol;

...

// Filtro para buscar todas as ações com a descrição ‘Petrobras’ da origem 12 (Bovespa)

SymbolSearchParam searchParam = new SymbolSearchParam();

searchParam.SourceId = "12";

searchParam.SymbolID = "";

searchParam.Description = "Petrobras";

searchParam.Market = "";

// Solicitar símbolos para o filtro desejado

cmaStreamerFeed.SymbolSearch(searchParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)).Dispose();

...

static void AsyncResultTestWriter(Result<List<SymbolSearchResult>> result)

{

if (result.Status.Success)

{

foreach (SymbolSearchResult symbol in result.Data)

{

Console.WriteLine($"\nSymbol consultado: " + symbol.ToJSON());

}

}

else

{

Console.WriteLine($"\nSymbol consultado retornou o status: " + result.Status.ToJSON());

}

}

# 2.4.3) Solicitação de Gráfico para API (HistoricalData)

▪ Método **HistoricalData**;

▪ Utilizar o método para uso ***Sem Assinatura***:

QuotesHistory(QuotesHistoryParam quoteshistoryParam, int millisecondsTimeOut = 10000)

▪ Utilizar o método para uso ***Com Assinatura (online)***:

QuotesHistory(QuotesHistoryParam quoteshistoryParam, bool sign, CancellationToken cancelToken, int millisecondsTimeOut = 10000, bool onlyDiff = false)

Ou

QuotesHistory(List<QuotesHistoryParam> quoteshistoryParamList, bool sign, CancellationToken cancelToken, int millisecondsTimeOut = 10000, bool onlyDiff = false)

Onde:

- sign = Utilizar true para assinatura;

- CancellationToken = Token que permite cancelar a assinatura (gerar token com o comando “*new CancellationTokenSource()*”).

*Atenção*: Utilizar o token gerado para efetuar o cancelamento da assinatura (comando “*cancellationTokenSource.Cancel();*”) que não está sendo mais utilizada, permitindo assim reduzir o consumo de memória, processador e rede.

▪ Parâmetros de entrada = Estrutura **QuotesHistoryParam** (quando usado assinatura é possível utilizar lista de símbolos utilizando coleção da estrutura):

⇨ VendorId [opcional] (integer) = Id do Vendor:

⬝ 1 (default) = RealTime;

⬝ 3 = SnapShot (foto do momento);

⬝ 4 = SnapShot End-Day (foto do final do dia);

⬝ 7 = Contribuidor;

⬝ 8 = DelayTime.

Observação: Vendor solicitado deve estar liberado no Advanced.

⇨ SourceID [obrigatório] (string) = Id da Origem do símbolo (exemplo: 12 – Bovespa);

⇨ SymbolID [obrigatório] (string) = Código da ação/opção/contrato/etc a ser pesquisada;

⇨ Period [obrigatório] (string) = Período desejado no gráfico:

- #m = Mínuto (exemplo: 1m);

- #h = Hora (exemplo: 1h);

- #d = Dia (exemplo: 1d);

- weekly = Semanal;

- monthly = Mensal;

- yearly = Anual.

⇨ DateFrom [obrigatório] (string) = Data início no formato YYYY-MM-DD HH:MM;

⇨ DateTo [obrigatório] (string) = Data final no formato YYYY-MM-DD HH:MM;

⇨ ClientMneMD (opcional) (string) = Mnemônico do Cliente OMS para contabilização de “*fee*” do Market Data;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

▪ Retorno = Estrutura **QuotesHistoryResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura QuotesHistoryResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com os pontos para o gráfico (pode retornar sem nenhum dado):

⬝ VendorId (integer) = Id do Vendor;

⬝ VendorDesc (string) = Descrição do Vendor;

⬝ Symbol:

- SourceId (string) = Bolsa de origem;

- SymbolID (string) = Código do símbolo (ticker da bolsa);

⬝ Date (string) = Data do ponto no formato YYYY-MM-DD HH:MM:SS;

⬝ Open (double) = Preço de abertura;

⬝ High (double) = Maior preço;

⬝ Low (double) = Menor preço;

⬝ Close (double) = Preço de fechamento;

⬝ AcumulatedQtty (double) = Quantidade acumulada;

⬝ Volume (double) = Volume financeiro.

# 2.4.4) Solicitação de Livro de Ofertas para API (Book)

▪ Método **Book**;

▪ Utilizar o método para uso ***Sem Assinatura***:

Book(BookParam bookParam, int millisecondsTimeOut = 10000)

▪ Utilizar o método para uso ***Com Assinatura (online)***:

Book(BookParam bookParam, bool sign, CancellationToken cancelToken = default, int millisecondsTimeOut = 10000)

Onde:

- sign = Utilizar true para assinatura;

- CancellationToken = Token que permite cancelar a assinatura (gerar token com o comando “*new CancellationTokenSource()*”).

*Atenção*: Utilizar o token gerado para efetuar o cancelamento da assinatura (comando “*cancellationTokenSource.Cancel();*”) que não está sendo mais utilizada, permitindo assim reduzir o consumo de memória, processador e rede.

▪ Parâmetros de entrada = Estrutura **BookParam**:

⇨ SourceID [obrigatório] (string) = Id da Origem do símbolo (exemplo: 12 – Bovespa);

⇨ Symbol [obrigatório] (string) = Código da ação/opção/contrato/etc;

⇨ Consolidate [obrigatório] (bool) = Se true consolida o livrio de ofertas por preço;

⇨ Deep [obrigatório] (string) = Profundidade do book (exemplo: “5” para cinco ofertas);

⇨ ClientMneMD (opcional) (string) = Mnemônico do Cliente OMS para contabilização de “*fee*” do Market Data;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

▪ Retorno = Estrutura **BookResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura BookResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com os dados do livro de ofertas (pode retornar sem nenhum dado):

⬝ Consolidate (bool) = Se true o livro de ofertas é consolidado por preço;

⬝ SymbolId:

- SourceID (string) = Bolsa de origem;

- SymbolID (string) = Código do símbolo (ticker da bolsa);

⬝ BuyOffers = Lista com as ofertas de compra:

- Agent (string) = Código da Corretora (se Consolidate = false);

- AgentName (string) = Nome da Corretora (se Consolidate = false);

- OfferCount (string) = Quantidade de ofertas (se Consolidate = true);

- Quantity (string) = Quantidade da oferta;

- Price (string) = Preço da oferta.

⬝ SellOffers = Lista com as ofertas de venda:

- Agent (string) = Código da Corretora (se Consolidate = false);

- AgentName (string) = Nome da Corretora (se Consolidate = false);

- OfferCount (string) = Quantidade de ofertas (se Consolidate = true);

- Quantity (string) = Quantidade da oferta;

- Price (string) = Preço da oferta.

# 2.4.5) Solicitação de Notícias para API (News)

▪ Método **News**;

▪ Utilizar o método para uso ***Sem Assinatura***:

News(NewsParam newsParam, int millisecondsTimeOut = 10000)

▪ Utilizar o método para uso ***Com Assinatura (online)***:

News(NewsParam newsParam, bool sign, CancellationToken cancelToken, int millisecondsTimeOut = 10000)

Onde:

- sign = Utilizar true para assinatura;

- CancellationToken = Token que permite cancelar a assinatura (gerar token com o comando “*new CancellationTokenSource()*”).

*Atenção*: Utilizar o token gerado para efetuar o cancelamento da assinatura (comando “*cancellationTokenSource.Cancel();*”) que não está sendo mais utilizada, permitindo assim reduzir o consumo de memória, processador e rede.

▪ Parâmetros de entrada = Estrutura **NewsParam**:

⇨ VendorId [opcional] (integer) = Id do Vendor:

⬝ 1 (default) = RealTime;

⬝ 3 = SnapShot (foto do momento);

⬝ 4 = SnapShot End-Day (foto do final do dia);

⬝ 7 = Contribuidor;

⬝ 8 = DelayTime.

Observação: Vendor solicitado deve estar liberado no Advanced.

⇨ SourceID [obrigatório] (int) = Id da Origem da notícia (exemplo: 12 – Bovespa);

⇨ Quantity [obrigatório] (int) = Número de notícia para retorno;

⇨ ImportantFilter [obrigatório] (bool) = Se true retorna apenas notícias importantes;

⇨ DateFilter [opcional] (string) = Data início no formato YYYY-MM-DD;

⇨ TextFilter [opcional] (string) = Filtro para o texto da notícia;

⇨ SubjectFilter [opcional] (string) = Filtro para o assunto da notícia;

⇨ ClientMneMD (opcional) (string) = Mnemônico do Cliente OMS para contabilização de “*fee*” do Market Data;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

▪ Retorno = Estrutura **NewsResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura RewsResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com os dados das notícias (pode retornar sem nenhum dado):

⬝ VendorId (integer) = Id do Vendor;

⬝ VendorDesc (string) = Descrição do Vendor;

⬝ Source (int) = Id da Origem da notícia;

⬝ Key (string) = Chave identificadora da notícia;

⬝ Sequence (int) = Número sequencial da notícia;

⬝ Date (string) = Data da notícia no formato YYYYMMDD;

⬝ Time (string) = Horário da notícia no formato HHMMSS;

⬝ Important (bool) = Se true a notícia é importante;

⬝ Subject (string) = Assunto da notícia;

⬝ Headline (string) = Título da notícia.

Atenção! Necessário ajustar as configurações do Streamer para efetuar “*cache*” das origens de notícias acessadas pela API, caso contrário a performance desta consulta não será adequada.

# 2.4.6) Solicitação da Íntegra da Notícia para API (News)

▪ Método **News**;

▪ Parâmetros de entrada = Estrutura **NewsBodyParam**:

⇨ SourceId [obrigatório] (int) = Id da Origem da notícia (exemplo: 12 – Bovespa);

⇨ Key [obrigatório] (string) = Número Chave da Notíca (campo Key recebido na NewsResult);

⇨ Sequence [opcional] (string) = Número Chave da Notíca (campo Sequence recebido na NewsResult);

⇨ Date [opcional] (string) = Número Chave da Notíca (campo Date recebido na NewsResult);

⇨ ClientMneMD (opcional) (string) = Mnemônico do Cliente OMS para contabilização de “*fee*” do Market Data;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

▪ Retorno = Estrutura **NewsBodyResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura NewsBodyResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com os dados das notícias (pode retornar sem nenhum dado):

⬝ Source (int) = Id da Origem da notícia;

⬝ Key (string) = Chave identificadora da notícia;

⬝ Sequence (int) = Número sequencial da notícia;

⬝ Date (string) = Data da notícia no formato YYYYMMDD;

⬝ Time (string) = Horário da notícia no formato HHMMSS;

⬝ Important (bool) = Se true a notícia é importante;

⬝ Subject (string) = Assunto da notícia;

⬝ Headline (string) = Título da notícia.

# 2.4.7) Solicitação de High Lights para API (Highlights)

▪ Método **Highlights**;

▪ Detalhes:

Highlights(HighlightsParam highLightsParam, bool sign, CancellationToken cancelToken, int millisecondsToRequestLooping = 300, int millisecondsTimeOut = 10000)

Onde:

- sign = Utilizar true para assinatura ou false para snapshot;

- cancelToken = Token para cancelar assinatura do relatório.

▪ Exemplos relatórios exemplos solicitados via CMA S4:

1 - Maiores Compradores/Vendedores por Contrato:



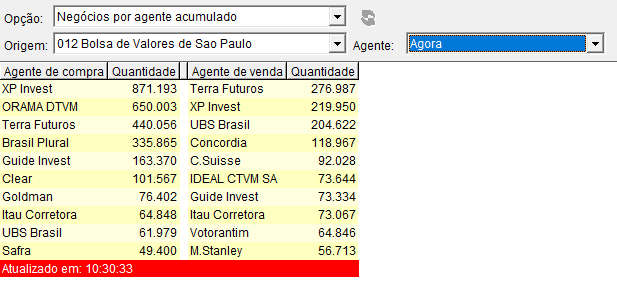
2 - Ativos Mais Negociados por Agente:



3 - Negócios por Agente:



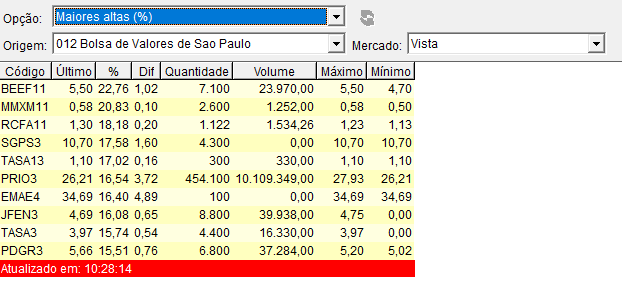
4 - Negócios por Agente Acumulado:



5 - Negócios Cruzados: Agente X Contrato:



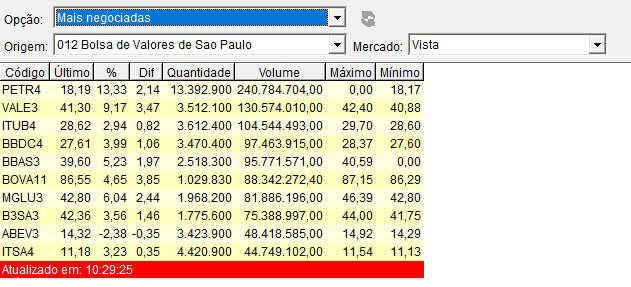
6 - Maiores Altas:



7 - Maiores Baixas:



8 - Mais Negociadas:



9 - Resumo de Negócios por Preço Médio:



▪ Parâmetros de entrada = Estrutura **HighlightsParam**:

⇨ Kind [obrigatório] (int) = Tipo de Relatório, segue lista:

1 - Maiores Compradores/Vendedores por Contrato;

2 - Ativos Mais Negociados por Agente;

3 - Negócios por Agente;

4 - Negócios por Agente Acumulado;

5 - Negócios Cruzados: Agente X Contrato;

6 - Maiores Altas;

7 - Maiores Baixas;

8 - Mais Negociadas;

9 - Resumo de Negócios por Preço Médio.

⇨ SourceId [obrigatório] (string) = Id da origem;

⇨ SymbolId [opcional] (string) = Código do Símbolo;

⇨ Market [opcional] (string) = Código do Mercado;

⇨ Broker [opcional] (string) = Código da Corretora;

⇨ Max [obrigatório] (int) = Número de Linhas;

⇨ SortingType [opcional] (int) = 1-Volume ou 2-Quantidade (se não informado assume 1);

⇨ DateFrom [opcional] (string) = Data inicial no formato YYYY-MM-DD (se não informado assume data atual);

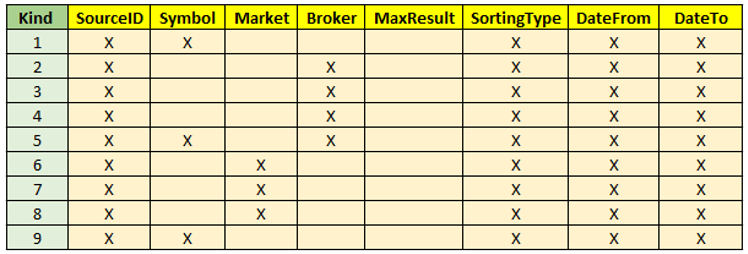
⇨ DateTo [opcional] (string) = Data final no formato YYYY-MM-DD (se não informado assume data atual);

⇨ ClientMneMD (opcional) (string) = Mnemônico do Cliente OMS para contabilização de “*fee*” do Market Data;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

▪ Tipos de Relatórios (Kind) vs Parâmetros:



▪ Retorno = Estrutura **HighlightsResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com os dados do relatório (pode retornar sem nenhum dado):

⬝ Report:

- Side (string) = Lado C-Compra ou V-Venda;

- Agent (string) = Corretora;

- Quantity (string) = Quantidade;

- Time (string) = Bolsa de origem;

- Symbol (string) = Código do símbolo;

- Market (string) = Mercado;

- Percent (string) = Bolsa de origem;

- Delta (string) = Delta;

- Maximum (string) = Preço máximo;

- Minimum (string) = Preço mínimo;

- Liquid (string) = Bolsa de origem;

- Buys (string) = Número de compras;

- Sells (string) = Número de vendas;

- Last (string) = Preço do último;

- AvgPrice (string) = Preço médio;

- AvgBuyPrice (string) = Preço médio de compra;

- AvgSellPrice (string) = Preço médio de venda.

⬝ LastUpdate (string) = Data da última atualização no formato YYYY-MM-DD HH:MM:SS.

# 2.5) Solicitação de Informações e Transações de Negociação para API

▪ As informações de Negociação são recebidas através dos servidores CMA Streamer conectados com os servidores CMA OMS (Order Manager System);

▪ O OMS é um sistema de negociação que permite negociar em várias bolsas do mundo e mercados diversificados, por este motivo a formatação de valores, preços e quantidade foram padronizados para atender estes vários mercados, segue detalhe:

⇨ Campos de valores são representados como string, sem separador de milhar, com separador de centavos usando “.” e com número de dígitos de centavos adequados ao mercado (exemplo: “*123456.78*”);

⇨ Campos de preços são representados como string, sem separador de milhar, com separador de decimais usando “.” e com número de dígitos de decimais adequados ao mercado (exemplo: “*1234.56*”);

⇨ Campos de quantidades são representados como string, sem separador de milhar, com separador de decimais usando “.” e com número de dígitos de decimais adequados ao mercado (exemplo: “*1234*”).

# 2.5.1) Solicitação de Lista de Contas OMS para a API (Client)

▪ Método **Client** com parâmetro **ClientListParam**;

▪ Parâmetros de entrada = Estrutura **ClientListParam**:

⇨ AssessorId [opcional] (long) = Id do Assessor;

⇨ BranchId [opcional] (long) = Id da Filial;

⇨ FilterType [obrigatório] (string) = Tipo do filtro (C=Código, D=Descrição e I=Documento);

⇨ FilterData [obrigatório] (string) = Valor do filtro;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

Observação: No caso do filterType=I-Documento o CPF/CNPJ será utilizado para a busca das contas do cliente (campo numérico sem marcadores e sem zeros a esquerda).

▪ Retorno = Estrutura **ClientListResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura ClientListResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com a lista de clientes (pode retornar sem nenhum cliente):

⬝ Id (long) = Id do Cliente no OMS (Id interno);

⬝ Name (string) = Nome do Cliente;

⬝ Mnemonic (string) = Mnemônico do Cliente no OMS;

⬝ Enabled (bool) = Cliente habilitado (true/false);

⬝ AccountTypeId (string) = Tipo de Conta do Cliente;

⬝ CategoryId (string) = Categoria de Conta;

⬝ BranchId (string) = Id da Filial (Id interno);

⬝ AssessorId (string) = Id do Assessor (Id interno);

⬝ OperatorId (string) = Id do Operador (Id interno);

⬝ BranchMne (string) = Mnemônico da Filial;

⬝ AssessorMne (string) = Mnemônico do Assessor;

⬝ OperatorMne (string) = Mnemônico do Operador;

⬝ CurrentAccountTypeId (string) = Tipo de Conta Corrente;

⬝ Email (string) = E-mail do Cliente.

▪ Exemplo assíncrono para solicitar a lista de contas OMS:

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Trading.ClientList;

...

// Filtro para buscar contas do cliente com CPF=12345678900

ClientListParam clientListParam = new ClientListParam();

clientListParam.FilterType = "I";

clientListParam.FilterData = "123456789";

// Solicitar lista de contas para o filtro desejado

cmaStreamerFeed.Client(clientListParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<List<ClientListResult>> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nLista de contas: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nLista de contas retornou o status: " + result.Status.ToJSON());

}

}

# 2.5.2) Solicitação de Inclusão de Ordem para API (Order ⇨ NewOrderSingleParam)

▪ Método **Order** com parâmetro **NewOrderSingleParam**;

▪ Parâmetros de entrada mais utilizados = Estrutura **NewOrderSingleParam**:

⇨ UserId = Identificação do cliente para o qual a ordem será atribuída:

⬝ User [obrigatório] (long) = Id numérico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Id*”);

⬝ UserMne [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⬝ Branch [opcional] (string) = Mnemônico da filial do cliente (obtido da estrutura “*ClientListResult*”).

⇨ TypeMarket [obrigatório] (string) = Id do mercado (utilizar 12010 para BVSP.VIS):

⬝ 9010 = BMF.FUT;

⬝ 9020 = BMF.OPD;

⬝ 9030 = BMF.OPF;

⬝ 9040 = BMF.TMO;

⬝ 9050 = BMF.DIS;

⬝ 12010 = BVSP.VIS;

⬝ 12020 = BVSP.OPC;

⬝ 12030 = BVSP.TRM.

⇨ Symbol [obrigatório] (string) = Símbolo do papel na Bolsa;

⇨ Side [obrigatório] (char) = B-Compra ou S-Venda;

⇨ Qtty [obrigatório] (string) = Quantidade da ordem;

⇨ Price [opcional] (string) = Preço da ordem;

⇨ TypeOffer [opcional] (string) = Tipo da ordem:

⬝ Market = Preço à mercado;

⬝ Limit (default) = Preço limitado ao valor “*Price*”;

⬝ StopMarket = Ao disparar o gatilho deve-se enviar uma ordem com preço à mercado;

⬝ StopLimit = Ao disparar o gatilho deve-se enviar uma ordem com o preço limitado ao valor “*Price*”.

⇨ TriggerType [opcional] (string) = Tipo de disparo:

⬝ None (default) = Ordem lançada assim que possível (se usado preço “*Limit*” preencher o campo “*Price*”);

⬝ Stop = Preço de gatilho no campo “*PriceTrigger*” (se usado preço “*Limit*” preencher o campo “*Price*”);

⬝ StopTrailing = Preço de gatilho móvel baseado no campo “*TraillingStop*” (se usado preço “*Limit*” preencher o campo “*Price*”).

⇨ PlaceDateType [opcional] (string) = Pregão que a ordem será enviada:

⬝ Today (default) = Pregão normal;

⬝ After = After-market;

⬝ Next = Próximo pregão.

⇨ TypeNegotiation [opcional] (string) = Tipo de negociação:

⬝ PDR (default) = Negociação padrão;

⬝ DTR = Negociação daytrade.

⇨ ValidityType [opcional] (string) = Tipo de validade;

⬝ Today (default) = Somente para a data atual;

⬝ UntilCancel = Até que seja pedido o cancelamento;

⬝ ImediateOrCancel = Executa imediatamente ou cancela;

⬝ Dated = Até a data determinada no campo “*Validity*”;

⬝ AllOrNothing = Ou toda a ordem é executada no instante de lançamento, ou cancela;

⬝ AtOpenning = Na abertura do mercado;

⬝ GoodForAuction = Boa para o leilão.

⇨ Validity [opcional] (string) = Data de validade no formato YYYY-MM-DD (deve ser preenchido quando “*ValidityType = Dated*”);

⇨ QuotesLastPrice [opcional] (string) = Preço da última cotação;

⇨ QuotesClosePrice [opcional] (string) = Preço do fechamento;

⇨ QuotesBidPrice [opcional] (string) = Preço da melhor oferta de compra;

⇨ QuotesAskPrice [opcional] (string) = Preço da melhor oferta de venda;

⇨ QuotesTime [opcional] (string) = Hora em que ocorreu a cotação;

⇨ IpRemote [opcional] (string) = Endereço ip do requisitante;

⇨ HostName [opcional] (string) = Nome do host do requisitante;

⇨ Confirmation [obrigatório] (bool) = Indicativo se o envio da ordem está sendo utilizado para montar a tela de confirmação (True - default) ou se a ordem deve ser gravada no OMS (False);

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

Observação: Se enviado “*UserMne*” (mnemônico do OMS) o campo “*User*” pode ser enviado com o valor 0 (zero).

▪ Demais parâmetros de entrada **NewOrderSingleParam**:

⇨ HideQtty [opcional] (string) = Quantidade aparente/aberta;

⇨ FractionaryPrice [opcional] (string) = Preço do símbolo a ser negociado no mercado fracionário;

⇨ PriceTrigger [opcional] (string) = Preço de disparo do gatilho;

⇨ PriceTriggerGain [opcional] (string) = Preço de disparo do gatilho de ganho;

⇨ PriceGain [opcional] (string) = Preço quando disparado gatilho de ganho;

⇨ PriceTriggerLoss [opcional] (string) = Preço de disparo do gatilho de perda;

⇨ PriceLoss [opcional] (string) = Preço de disparado gatilho de perda;

⇨ TraillingStop [opcional] (string) = Margem de elasticidade do gatilho móvel;

⇨ ExDestination [opcional] (string) = Código do roteamento;

⇨ Isin [opcional] (string) = Número de identificação internacional de valor mobiliários;

⇨ AccountId [opcional] (long) = Código da conta do usuário a qual será efetuado o débito;

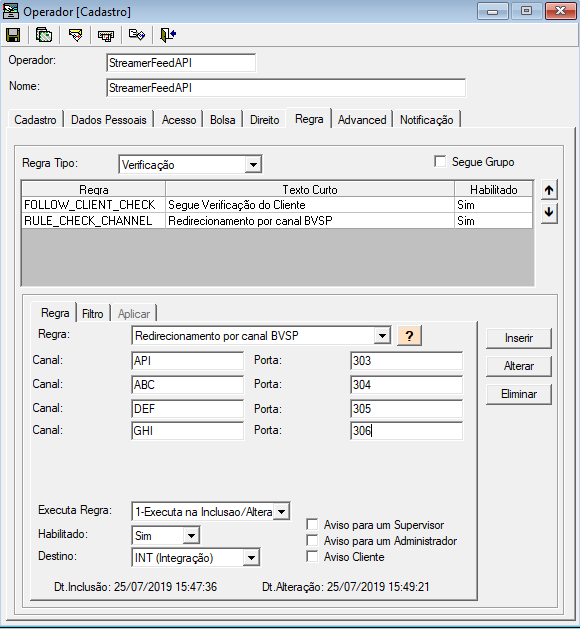
⇨ AgreementSign [opcional] (string) = Tipo do termo Suitability assinado pelo usuário;

⇨ Justification [opcional] (string) = Justificativa de desenquadramento realizada pelo operado;

⇨ Identification [opcional] (string) = Identificação atribuída pelo usuário no envio da ordem;

⇨ Signature [opcional] (string) = Assinatura eletrônica (se configurado no Streamer);

⇨ TradeChannel [opcional] (string) = Canal de negociação para redirecionamento da integração via regra de verificação “*Redirecionamento por Canal BVSP*” (vide item #11 do documento CMAeBroker\_Regras.doc). Atenção! O canal de negociação não sobrescreve o canal enviado no momento da criação da instância da API, ou seja, a ordem continua sendo alocada para o canal enviado na criação da API e o canal enviado no NewOrderSingle será utilizado exclusivamente pela regra “*Redirecionamento por Canal BVSP*” que precisa estar cadastrada e configurada, segue imagem exemplo da regra:



⭢ O canal do TradeChannel não precisa ser um canal cadastrado no OMS (somente será utilizado dentro desta regra).

▪ Retorno = Estrutura **NewOrderSingleResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura NewOrderSingleResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Dados da ordem:

⬝ OrderId (string): Id da ordem no OMS;

⬝ FractionaryOrderId (string): Id da ordem no OMS (enviada somente quando o envio de ordem com geração de ordens no mercado a Vista e Fracionário);

⬝ Symbol = Dados do símbolo:

- SourceId (string) = Id do mercado OMS (exemplo: 12010 para BVSP);

- SymbolId (string) = Código do símbolo.

⬝ ContractDescription (string) = Descrição do símbolo;

⬝ AddDate (string) = Data da inclusão no formato YYYY-MM-DD;

⬝ UserId = Identificação do cliente:

- User (string) = Id numérico do cliente;

- UserMne (string) = Mnemônico do cliente;

- Branch (string) = Id da filial.

⬝ Side (char) = B-Compra ou S-Venda;

⬝ Qtty (string) = Quantidade;

⬝ HideQtty (string) = Quantidade aparente/aberta;

⬝ Value (string) = Valor;

⬝ LimitPrice (string) = Preço limite;

⬝ TriggerPrice (string) = Preço de disparo do Stop;

⬝ TrailingStop (string) = % de acompanhamento do Stop Móvel;

⬝ TriggerGain (string) = Preço de disparo do Stop Ganho;

⬝ PriceGain (string) = Preço limite para Stop Ganho;

⬝ TriggerLoss (string) = Preço de disparo do Stop Perda;

⬝ PriceLoss (string) = Preço limite para Stop Perda;

⬝ PendingQtty (string) = Quantidade pendente;

⬝ ProcessedDate (string) = Data do envio para a bolsa no formato YYYY-MM-DD;

⬝ ExpiredQtty (string) = Quantidade expirada;

⬝ ExpireDate (string) = Data da expiração no formato YYYY-MM-DD;

⬝ ConfirmedQtty (string) = Quantidade confirmada (negócio);

⬝ ConfirmedDate (string) = Data da confirmação no formato YYYY-MM-DD;

⬝ ConfirmedPrice (string) = Preço confirmação (negócio);

⬝ ConfirmedValue (string) = Valor confirmação (negócio);

⬝ CancelQtty (string) = Quantidade cancelada;

⬝ CancelDate (string) = Data do cancelamento no formato YYYY-MM-DD;

⬝ Cancelable (bool) = Se true a ordem permite cancelamento;

⬝ CanAlter (bool) = Se true a ordem permite alteração;

⬝ OfferType (string) = Tipo da oferta:

- NOR = Normal (lançada assim que possível para a bolsa);

- STOP = Stop (disparada quando atingir o preço de disparo);

- STOP\_TR = Stop Móvel (disparada quando atingir o preço de disparo);

- STOP\_GL = Stop Gain/Loss (disparada quando atingir o preço de disparo).

⬝ PriceType (string) = Tipo do preço:

- LIM = Preço limite (preço preenchido nos campos LimitPrice, TriggerPrice, PriceGain ou PriceLoss);

- MKT = Ordem com preço a mercado (preço não informado);

- AOP = Ordem com preço de leilão (preço não informado).

⬝ ValidityType (string) = Tipo da validade:

- Dated = Data de validade no campo ValidateDate;

- VAC = Válida até cancelamento;

- EC = Executa ou cancela;

- TN = Tudo ou nada.

⬝ Integral (bool) = Se true indica que a ordem permite apenas negócio integral;

⬝ Generation (string) = Identifica a origem da ordem (se foi gerada pelo Usuário, Alarme ou Trading System);

⬝ Identification (string) = Identificação atribuída pelo usuário no envio da ordem;

⬝ ClientId (string) = Id do cliente da ordem no OMS;

⬝ BrokerClientId (string) = Conta do cliente na bolsa;

⬝ BrokerOperatorMne (string) = Mnemônico do operador OMS;

⬝ BrokerOperatorName (string) = Nome do operador OMS;

⬝ BrokerOperatorBackoffice (string) = Nome do operador back-office;

⬝ ValidateDate (string) = Data de validade no formato YYYY-MM-DD;

⬝ Modality (string) = Modalidade da ordem:

- Today = Ordem a ser inserida no pregão normal;

- After = Ordem a ser inserida no pregão after-market;

- Next = Ordem a ser inserida no próximo pregão normal.

⬝ MarketId (string) = Id do mercado no OMS (exemplo: 12010 BVSP.VIS);

⬝ TypeMode (string) = Tipo de modalidade:

- VIS = Vista/Lote;

- FRA = Fracionario.

⬝ TriggeredDate (string) = Data do disparo do Stop no formato YYYY-MM-DD HH:MM:SS;

⬝ TriggeredQuote (string) = Cotação que ocasionou o disparo;

⬝ Message (string) = Mensagem adicionada a ordem;

⬝ ExDestination (string) = Ex-destination;

⬝ ExchangeNumber (string) = Número da ordem na bolsa;

⬝ ExchangeNumberDetail (string) = Detalhes do número da ordem na bolsa;

⬝ CurrentAccountTypeId (string) = Tipo de Conta Corrente;

⬝ OperatorTypeId (string) = Tipo do Operador;

⬝ ChannelId (string) = Canal de inclusão da ordem;

⬝ MovementDate (string) = Data do pregão no formato YYYY-MM-DD;

⬝ Scheduled (bool) = Se true indica que a ordem está agendada;

⬝ TypeNegotiation (string) = Tipo de negociação:

- PDR = Negociação padrão;

- DTR = Negociação daytrade.

⬝ Costs = Custos da ordem:

- Brokerage (string) = Valor de corretagem;

- Fee (string) = valor de emolumentos da bolsa;

- Registry (string) = Valor de registro na bolsa.

⬝ CostsDayTrade = Custos do day-trade:

- Brokerage (string) = Valor de corretagem;

- Fee (string) = valor de emolumentos da bolsa;

- Registry (string) = Valor de registro na bolsa.

⬝ FinancialLiquidation (string) = Data liquidação financeira no formato YYYY-MM-DD;

⬝ PositionLiquidation (string) = Data liquidação custódia no formato YYYY-MM-DD;

⬝ ProcessingDate (string) = Data envio bolsa no formato YYYY-MM-DD HH:MM:SS;

⬝ TypeSuitability (string) = Tipo do termo Suitability a ser mostrado ao cliente;

⬝ AlertText (string) = Mensagem de alerta Suitability;

⬝ AgreementText (string) = Mensagem para o termo de aceite;

⬝ Profile (string) = Perfil do investidor;

⬝ NewProfile (string) = Novo perfil do investidor considerando esta nova ordem;

⬝ SymbolText (string) = Mensagem cadastrada para o símbolo.

▪ Enviando ordem para duas modalidades diferentes Vista (integral) e Fracionária:

⇨ Preencher “*NewOrderSingleParam*” o campo “*Qtty*” com a quantidade total (exemplo: quantidade de 150 PETR4 que deve gerar 100 na modalidade Vista e 50 na Fracionária);

⇨ Se o preço da modalidade Fracionária for diferente é necessário preencher o campo “*FractionaryPrice*”;

⇨ Na resposta ”*NewOrderSingleResult*” o campo “*FractionaryOrderId*” será preenchido com o ID da ordem na modalidade fracionária.

▪ Exemplo assíncrono para inclusão de ordem para o cliente ID-OMS 68975:

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Trading.Order.OrderSingle;

...

// Parâmetros para inclusão da ordem

OrderSingleParam orderSingleParam = new OrderSingleParam();

orderSingleParam.UserId.User = 68975;

orderSingleParam.Symbol = "BBDC4";

orderSingleParam.Side = "B";

orderSingleParam.Qtty = "500";

orderSingleParam.Price = "17.50";

orderSingleParam.TypeOffer = "Limit";

orderSingleParam.TriggerType = "None";

orderSingleParam.ValidityType = "Today";

orderSingleParam.Validity = "2018-07-23";

orderSingleParam.TypeMarket = "12010";

orderSingleParam.Confirmation = false;

orderSingleParam.PlaceDateType = "Today";

orderSingleParam.QuotesLastPrice = "17.48";

orderSingleParam.QuotesClosePrice = "17.48";

orderSingleParam.QuotesBidPrice = "17.50";

orderSingleParam.QuotesAskPrice = "17.52";

orderSingleParam.QuotesTime = "16:01";

orderSingleParam.IpRemote = "10.0.13.16";

orderSingleParam.HostName = "spo-dev01.cmna.com.br";

// Solicitar inclusão da ordem

cmaStreamerFeed.Order(orderSingleParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<OrderSingleResult> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nInclusão da ordem: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nSolicitação de inclusão de ordem retornou o status: " + result.Status.ToJSON());

}

}.

# 2.5.3) Solicitação de Cancelamento de Ordem para API (Order ⇨ OrderCancelParam)

▪ Método **Order** com parâmetro **OrderCancelParam**;

▪ Parâmetros de entrada = Estrutura **OrderCancelParam**:

⇨ UserId = Identificação do cliente da ordem a ser cancelada:

⬝ User [obrigatório] (long) = Id numérico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Id*”);

⬝ UserMne [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⬝ Branch [opcional] (string) = Mnemônico da filial do cliente (obtido da estrutura “*ClientListResult*”).

⇨ OfferId [obrigatório] (string) = Número da ordem;

⇨ IpRemote [opcional] (string) = Endereço ip do requisitante;

⇨ HostName [opcional] (string) = Nome do host do requisitante;

⇨ Signature [opcional] (string) = Assinatura eletrônica (se configurado no Streamer);

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

Observação: Se enviado “*UserMne*” (mnemônico do OMS) o campo “*User*” pode ser enviado com o valor 0 (zero).

▪ Retorno = Estrutura **OrderCancelResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura OrderCancelResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Não utilizado.

▪ Exemplo assíncrono para cancelamento da ordem Id = 1:

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1.Trading.Order.OrderCancel;

...

// Parâmetros para solicitar o cancelamento da ordem

OrderCancelParam orderCancelParam = new OrderCancelParam();

orderCancelParam.UserId.User = 68975;

orderCancelParam.OfferId = "1365";

orderCancelParam.IpRemote = "10.0.13.16";

orderCancelParam.HostName = "spo-dev01.cma.com.br";

// Solicitar cancelamento da ordem

cmaStreamerFeed.Order(orderCancelParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<OrderCancelResult> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nCancelamento da ordem solicitado: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nSolicitação de cancelamento da ordem retornou o status: " + result.Status.ToJSON());

}

}

# 2.5.4) Solicitação de Alteração de Ordem para API (Order ⇨ OrderReplaceParam)

▪ Método **Order** com parâmetro **OrderReplaceParam**;

▪ Parâmetros de entrada = Estrutura **OrderReplaceParam**:

⇨ UserId = Identificação do cliente da ordem que será alterada:

⬝ User [obrigatório] (long) = Id numérico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Id*”);

⬝ UserMne [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⬝ Branch [opcional] (string) = Mnemônico da filial do cliente (obtido da estrutura “*ClientListResult*”).

⇨ OfferId [obrigatório] (string) = Número da ordem;

⇨ TypeOffer [obrigatório] (string) = Tipo da ordem:

⬝ Market = Preço à mercado;

⬝ Limit = Preço limitado ao valor “Price”;

⬝ StopMarket = Ao disparar o gatilho deve-se enviar uma ordem com preço à mercado;

⬝ StopLimit = Ao disparar o gatilho deve-se enviar uma ordem com o preço limitado ao valor “Price”.

⇨ TypeValidity [obrigatório] (string) = Tipo de validade:

⬝ Today = Somente para a data atual;

⬝ UntilCancel = Até que seja pedido o cancelamento;

⬝ ImediateOrCancel = Executa imediatamente ou cancela;

⬝ Dated = Até a data determinada no campo “Validity”;

⬝ AllOrNothing = Ou toda a Oferta é executada no instante de lançamento, ou cancela;

⬝ AtOpenning = Na Abertura do Mercado;

⬝ GoodForAuction = Boa para o leilão.

⇨ Validity [opcional] (string) = Data de validade no formato YYYY-MM-DD (quando “ValidityType = Dated”);

⇨ OriginalQtty [obrigatório] (string) = Quantidade atual da ordem (antes da alteração);

⇨ Qtty [opcional] (string) = Nova quantidade da ordem;

⇨ Price [opcional] (string) = Novo preço da ordem;

⇨ HideQtty [opcional] (string) = Nova quantidade aparente/aberta;

⇨ QuotesLastPrice [opcional] (string) = Preço da última cotação;

⇨ QuotesClosePrice [opcional] (string) = Preço do fechamento;

⇨ QuotesBidPrice [opcional] (string) = Preço da melhor oferta de compra;

⇨ QuotesAskPrice [opcional] (string) = Preço da melhor oferta de venda;

⇨ QuotesTime [opcional] (string) = Hora em que ocorreu a cotação;

⇨ IpRemote [opcional] (string) = Endereço ip do requisitante;

⇨ HostName [opcional] (string) = Nome do host do requisitante;

⇨ Confirmation [opcional] (bool) = Indicativo se o envio da ordem está sendo utilizado para montar a tela de confirmação (True) ou se a alteração deve ser gravada no OMS (False);

⇨ Signature [opcional] (string) = Assinatura eletrônica (se configurado no Streamer);

⇨ TradeChannel [opcional] (string) = Vide informações no “*NewOrderSingleParam*”;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

Observação: Se enviado “*UserMne*” (mnemônico do OMS) o campo “*User*” pode ser enviado com o valor 0 (zero).

▪ Retorno = Estrutura **OrderReplaceResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura OrderreplaceResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Dados da alteração:

⬝ TypeSuitability (string) = Tipo do termo Suitability a ser mostrado ao cliente;

⬝ AlertText (string) = Mensagem de alerta Suitability;

⬝ AgreementText (string) = Mensagem para o termo de aceite;

⬝ Profile (string) = Perfil do investidor;

⬝ NewProfile (string) = Novo perfil do investidor considerando esta nova ordem.

▪ Exemplo assíncrono para alteração da ordem Id = 1:

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1.Trading.Order.OrderReplace;

...

// Parâmetros para solicitar a alteração da ordem

OrderReplaceParam orderReplaceParam = new OrderReplaceParam();

orderReplaceParam.UserId.User = 68975;

orderReplaceParam.TypeOffer = "Limit";

orderReplaceParam.OfferId = "1365";

orderReplaceParam.Qtty = "200";

orderReplaceParam.Price = "17.50";

orderReplaceParam.OriginalQtty = "500";

orderReplaceParam.TypeValidity = "Today";

orderReplaceParam.Validity = "20180725";

orderReplaceParam.AcceptPartial = true;

orderReplaceParam.HideQtty = "0";

orderReplaceParam.Confirmation = false;

orderReplaceParam.QuotesLastPrice = "17.60";

orderReplaceParam.QuotesClosePrice = "17.48";

orderReplaceParam.QuotesBidPrice = "17.49";

orderReplaceParam.QuotesAskPrice = "17.52";

orderReplaceParam.QuotesTime = "16:03:09";

orderReplaceParam.IpRemote = "10.0.13.16";

orderReplaceParam.HostName = "spo-dev01.cma.com.br";

// Solicitar alteração da ordem

cmaStreamerFeed.Order(orderReplaceParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<OrderReplaceResult> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nAlteração da ordem solicitada: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nSolicitação de alteração da ordem retornou o status: " + result.Status.ToJSON());

}

}

# 2.5.5) Solicitação de Lista de Ordens para API (Order ⇨ OrderListParam)

▪ Método **Order** com parâmetro **OrderListParam**;

Order(OrderListParam orderListParam, int millisecondsTimeOut = 10000)

▪ Parâmetros de entrada = Estrutura **OrderListParam**:

⇨ ClientIdFilter [obrigatório] (long) = Identificação do cliente para a lista de ordens (obtido da estrutura “*ClientListResult*” campo “*Id*”);

⇨ ClientMneFilter [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⇨ MarketIdFilter [opcional] (long) = Id do Mercado (exemplo: 12010 para BVSP.VIS);

⇨ MarketMneFilter [opcional] (string) = Mnemônico do Mercado (exemplo: BVSP.VIS);

⇨ PrefixFilter [opcional] (string) = Mnemônico do Prefixo (exemplo: WIN.FUT);

⇨ OperationTypeFilter [opcional] (string) = B-Compra ou S-Venda (default ambas);

⇨ OfferStatusFilter [opcional] (string) = Situações das ofertas (default “PECXV"):

⬝ P = Ordens Pendentes;

⬝ E = Ordens Processadas;

⬝ C = Ordens Confirmadas (négocio);

⬝ X = Ordens Canceladas;

⬝ V = Ordens Vencidas;

⬝ T = Ordens Stop não disparadas.

⇨ FilterCancelRequested [opcional] (string) = Ordens com cancelamento solicitado:

⬝ F = Lista apenas ordens com requisição de cancelamento;

⬝ S = Lista ordens com requisição de cancelamento como sendo situação;

⬝ N (default) = Lista ordens independente da situação de requisição de cancelamento.

⇨ FilterCancelRejected [opcional] (string) = Ordens com cancelamento solicitado e rejeitado:

⬝ F = Lista apenas ordens com rejeição de cancelamento;

⬝ S = Lista ordens com rejeição de cancelamento como sendo situação;

⬝ N (default) = Lista ordens independente da situação de rejeição de cancelamento.

⇨ DaysToList [opcional] (int) = Número de dias das ordens:

⬝ 1 (default) = Hoje;

⬝ 3 = Três últimos dias (corridos);

⬝ 7 = Sete últimos dias (corridos);

⬝ 0 = Usar o campo “*StartDate*”.

⇨ StartDate [opcional] (string) = Data inicial no formato YYYY-MM-DD (para DaysToList=0);

⇨ EndDate [opcional] (string) = Data final no formato YYYY-MM-DD (para DaysToList=0);

⇨ OperatorIdFilter [opcional] (long) = Id do Operador;

⇨ OperatorMneFilter [opcional] (string) = Mnemônico do Operador;

⇨ ListType [opcional] (string) = Tipo de Lista G-Agrupada ou F-Desagrupada Ordens-Mães/Filhas (default “*G*”);

⇨ Identificator [opcional] (string) = Identificação atribuída pelo usuário no envio da ordem;

⇨ TriggeredBy [opcional] (string) = Identificação da origem da ordem (se foi gerada pelo Usuário, Alarme ou Trading System);

⇨ OnlySelectedClients [opcional] (int) = Lista de favoritos:

⬝ 0 (default) = Lista as ordens de todos clientes ligados ao cliente solicitante;

⬝ 1 = Lista as ordens dos clientes selecionados pelo cliente solicitante;

⬝ 2 = Lista as ordens dos clientes selecionados pelo cliente solicitante e do próprio;

⬝ 3 = Lista as ordens de todos clientes ligados ao cliente do filtro.

⇨ SearchHistory [opcional] (bool) = Se true pesquisa no histórico (o uso de histórico pode causar lentidão na resposta por isso recomendamos utilizar false);

⇨ OfferMasterId [opcional] (string) = Id da ordem principal (mãe);

⇨ TypeOrderBy [opcional] (int) = Tipo de ordenação da lista:

⬝ 0 (default) = Id da ordem descendente;

⬝ 1 = Mercado ascendente;

⬝ 2 = Mercado descendente;

⬝ 3 = Tipo de operação ascendente;

⬝ 4 = Tipo de operação descendente;

⬝ 5 = Situação ascendente;

⬝ 6 = Situação descendente.

⇨ AddingDateConsult [opcional] (int) = Tipo da consulta da data:

⬝ 1 (default) = Data atualização da ordem;

⬝ 2 = Data de inclusão da ordem.

⇨ TypeOffer [opcional] (string) = Tipo da ordem:

⬝ Market = Preço à mercado;

⬝ Limit = Preço limitado ao valor “*Price*”;

⬝ StopMarket = Ao disparar o gatilho deve-se enviar uma ordem com preço à mercado;

⬝ StopLimit = Ao disparar o gatilho deve-se enviar uma ordem com o preço limitado ao valor “*Price*”.

⇨ MaxResults [opcional] (int) = Número de ordens retornadas (default 50);

⇨ ContractFilter [opcional] (string) = Símbolo;

⇨ OfferStatusFilterAND [opcional] (string) = Filtro com situação para “*and*” (caso contrário vai utilizar “or”);

⇨ OfferIdFilter1 [opcional] (string) = Id da ordem 1ª;

⇨ OfferIdFilter2 [opcional] (string) = Id da ordem 2ª;

⇨ OfferIdFilter3 [opcional] (string) = Id da ordem 3ª;

⇨ TypeMode [opcional] (string) = Tipo do modo da ordem (VIS-Vista ou FRA-Fracionário);

⇨ Scheduled [opcional] (int) = Filtro para ofertas agendadas:

⬝ 0 (default) = Não efetuar o filtro;

⬝ 1 = Apenas ordens agendadas;

⬝ 2 = Apenas ordens não agendadas.

⇨ OperatorTypeId [opcional] (long) = Id do operador;

⇨ ChannelId [opcional] (long) = Id do canal;

⇨ Triggered [opcional] (int) = Listar as ofertas do tipo "Stop":

⬝ 0 (default) = Qualquer ordem;

⬝ 1 = Somente ordens “stop”;

⬝ 2 = Somente ordens não “stop”.

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

Observação: Se enviado “*ClientMneFilter*” (mnemônico do OMS) o campo “*ClientIdFilter*” pode ser enviado com o valor 0 (zero).

▪ Retorno = Estrutura **OrderListResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura OrderListResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com a lista das ordens (pode retornar sem nenhuma ordem):

⬝ OrderId (string): Id da ordem no OMS;

⬝ SymbolId = Dados do símbolo:

- SourceId (string) = Id do mercado OMS (exemplo: 12010 para BVSP);

- Symbol (string) = Código do símbolo.

⬝ ContractDescription (string) = Descrição do símbolo;

⬝ AddDate (string) = Data da inclusão no formato YYYY-MM-DD;

⬝ UserId = Identificação do cliente:

- User (string) = Id numérico do cliente;

- UserMne (string) = Mnemônico do cliente;

- Branch (string) = Id da filial.

⬝ Side (char) = B-Compra ou S-Venda;

⬝ Qtty (string) = Quantidade da ordem;

⬝ HideQtty (string) = Quantidade aparente/aberta;

⬝ Status (string) = Situação da ordem (ListType=G para “*Agrupada*” pode conter múltiplas situações):

|  |  |  |
| --- | --- | --- |
| Status | Situação | Descrição |
| P | Pendente | Ordem processada pelo OMS mas não enviada para a bolsa. |
| E | Processada | Ordem processada pelo OMS e enviada para a bolsa. |
| C | Confirmada | Ordem confirmada (com negócio na bolsa). |
| X | Cancelada | Ordem cancelada. |
| T | Condicional | Ordem Stop ainda não disparada. |
| V | Vencida | Ordem que não foi executada até seu prazo de validade e venceu. |
| S | Agendada | Ordem agendada ainda não disparada por não ter sido atingida a data de disparo da ordem. |
| EC | Parcialmente Confirmada | Ordem confirmada parcialmente com resto pendente na bolsa. |
| ECX | Parcialmente Confirmada | Ordem confirmada parcialmente, com negócio cancelado pela bolsa, e resto pendente na bolsa. |
| EX | Parcialmente Cancelada | Ordem com negócio cancelado pela bolsa e com resto pendente na bolsa. |
| CV | Parcialmente Confirmada | Ordem confirmada parcialmente e o restante da ordem venceu sendo retirada da bolsa. |
| CX | Parcialmente Confirmada | Ordem confirmada parcialmente e o restante da ordem foi cancelada. |
| CXV | Parcialmente Confirmada | Ordem confirmada parcialmente com negócio cancelado pela bolsa e o restante da ordem venceu sendo retirada da bolsa. |
| XV | Parcialmente Cancelada | Ordem com negócio cancelado pela bolsa e o restante da ordem venceu sendo retirada da bolsa. |

Observação: Na ListType=G (lista agrupada) recomendamos utilizar “Parcialmente Confirmada” (ou Executada) quando múltiplos status contendo “C” e “Parcialmente Cancelada” quando múltiplos status contendo “X”.

⬝ Value (string) = Valor (ListType=G valor da ordem - ListType=F valor para o status);

⬝ LimitPrice (string) = Preço limite;

⬝ TriggerPrice (string) = Preço de disparo do Stop;

⬝ TrailingStop (string) = % de acompanhamento do Stop Móvel;

⬝ TriggerGain (string) = Preço de disparo do Stop Ganho;

⬝ PriceGain (string) = Preço limite para Stop Ganho;

⬝ TriggerLoss (string) = Preço de disparo do Stop Perda;

⬝ PriceLoss (string) = Preço limite para Stop Perda;

⬝ PendingQtty (string) = Quantidade pendente;

⬝ ProcessedDate (string) = Data do envio para a bolsa no formato YYYY-MM-DD HH:MM:SS;

⬝ ExpiredQtty (string) = Quantidade espirada;

⬝ ExpireDate (string) = Data da expiração no formato YYYY-MM-DD HH:MM:SS;

⬝ ConfirmedQtty (string) = Quantidade confirmada (negociada);

⬝ ConfirmedDate (string) = Data da confirmação no formato YYYY-MM-DD HH:MM:SS;

⬝ ConfirmedPrice (string) = Preço confirmação (negócio);

⬝ ConfirmedValue (string) = Valor confirmação (negócio);

⬝ CancelQtty (string) = Quantidade cancelada;

⬝ CancelDate (string) = Data do cancelamento no formato YYYY-MM-DD HH:MM:SS;

⬝ Cancelable (bool) = Se true a ordem permite cancelamento;

⬝ CanAlter (bool) = Se true a ordem permite alteração;

⬝ OfferType (string) = Tipo da oferta:

- NOR = Normal (lançada assim que possível para a bolsa);

- STOP = Stop (disparada quando atingir o preço de disparo);

- STOP\_TR = Stop Móvel (disparada quando atingir o preço de disparo);

- STOP\_GL = Stop Gain/Loss (disparada quando atingir o preço de disparo).

⬝ PriceType (string) = Tipo do preço:

- LIM = Preço limite (preço preenchido nos campos LimitPrice, TriggerPrice, PriceGain ou PriceLoss);

- MKT = Ordem com preço a mercado (preço não informado);

- AOP = Ordem com preço de leilão (preço não informado).

⬝ ValidityType (string) = Tipo da validade:

- Dated = Data de validade no campo ValidateDate;

- VAC = Válida até cancelamento;

- EC = Executa ou cancela;

- TN = Tudo ou nada.

⬝ Integral (bool) = Se true indica que a ordem permite apenas negócio integral;

⬝ Identification (string) = Identificação atribuída pelo usuário no envio da ordem;

⬝ ClientId (string) = Id do cliente que inseriu a ordem no OMS;

⬝ BrokerClientId (string) = Conta do cliente na bolsa;

⬝ BrokerOperatorMne (string) = Mnemônico do operador OMS;

⬝ BrokerOperatorName (string) = Nome do operador OMS;

⬝ BrokerOperatorBackoffice (string) = Nome do operador back-office;

⬝ ValidateDate (string) = Data de validade no formato YYYY-MM-DD;

⬝ MarketId (string) = Id do mercado no OMS (exemplo: 12010 BVSP.VIS);

⬝ TypeMode (string) = Tipo de modalidade:

- VIS = Vista/Lote;

- FRA = Fracionario.

⬝ TriggeredDate (string) = Data do disparo do Stop no formato YYYY-MM-DD HH:MM:SS;

⬝ TriggeredQuote (string) = Cotação que ocasionou o disparo;

⬝ Message (string) = Mensagem adicionada a ordem;

⬝ ExDestination (long) = Ex-destination;

⬝ ExchangeNumber (string) = Número da ordem na bolsa;

⬝ ExchangeNumberDetail (string) = Detalhes do número da ordem na bolsa;

⬝ OperatorTypeId (string) = Tipo do Operador;

⬝ ChannelId (string) = Canal de inclusão da ordem;

⬝ MovementDate (string) = Data do pregão no formato YYYY-MM-DD;

⬝ Scheduled (bool) = Se true indica que a ordem está agendada;

⬝ TypeNegotiation (string) = Tipo de negociação:

- PDR = Negociação padrão;

- DTR = Negociação daytrade.

⬝ SymbolText (string) = Mensagem cadastrada para o símbolo.

▪ Os campos para “*ListType*” G-Agrupada ou F-Ordens-Mães/Filhas são preenchidos diferente, segue detalhes:

⇨ “*ListType=F*” os campos “*PendingQtty*”, “*ExpiredQtty*”, “*ConfirmedQtty*” e “*CancelQtty*” não são preenchidos pois o campo no “*Qtty*” temos a quantidade da situação (campo “*Status*”);

⇨ “*ListType=F*” o campo “*LimitPrice*” é o preço original da ordem, no caso de “*Status=C*” o campo “*ConfirmedPrice*” informa o preço da confirmação/negócio;

⇨ “*ListType=F*” o campo “*OrderId*” é preenchido com o ID-Ordem-OMS seguido de “.” e o número sequencial do evento (exemplo: “*001331.1*”).

▪ Para obter todos os eventos de uma determinada ordem chamar com os parâmetros:

⇨ “*ListType=F*”;

⇨ “*OfferIdFilter1=n*” (onde “*n*” é o ID-Ordem-OMS que desejamos listar todos os eventos);

⇨ Também é possível filtrar obtendo somente as confirmações/negócios com o parâmetro “*Status=C*”;

⇨ Necessário cuidado com o filtro de período, pois uma ordem pode ter gerado eventos em vários dias.

▪ Exemplo assíncrono para solicitação da lista de ordem do cliente ID-OMS 68975:

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1.Trading.Order.OrderList;

...

// Parâmetros para solicitar a lista de ordens

OrderListParam orderListParam = new OrderListParam();

orderListParam.ClientIdFilter = 68975;

orderListParam.DaysToList = 7;

orderListParam.SearchHistory = false;

orderListParam.ListType = "G";

orderListParam.OfferStatusFilter = "PECXV";

// Solicitar a lista de ordens

cmaStreamerFeed.Order(orderListParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<List<OrderListResult>> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nLista de ordens: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nSolicitação de lista de ordens retornou o status: " + result.Status.ToJSON());

}

}

# 2.5.6) Solicitação de Portfólio (Custódia) para API (Portfolio)

▪ Método **Portfolio** com parâmetro **PortfolioParam**;

▪ Parâmetros de entrada = Estrutura **PortfolioParam**:

⇨ UserId = Identificação do cliente para a posição:

⬝ User [obrigatório] (long) = Id numérico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Id*”);

⬝ UserMne [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⬝ Branch [opcional] (string) = Mnemônico da filial do cliente (obtido da estrutura “*ClientList*”).

⇨ MarketId [opcional] (int) = Id do mercado (utilizar 12010 para BVSP.VIS):

⬝ 9010 = BMF.FUT;

⬝ 9020 = BMF.OPD;

⬝ 9030 = BMF.OPF;

⬝ 9040 = BMF.TMO;

⬝ 9050 = BMF.DIS;

⬝ 12010 = BVSP.VIS;

⬝ 12020 = BVSP.OPC;

⬝ 12030 = BVSP.TRM.

⇨ PortfolioType [obrigatório] (int) = Tipo da Carteira 1-Abertura ou 2-Provisória;

⇨ Referer [opcional] (objeto) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⇨ EnableTrace [opcional] (bool) = Vide item 2.3.1.

Observação: Se enviado “*UserMne*” (mnemônico do OMS) o campo “*User*” pode ser enviado com o valor 0 (zero).

▪ Retorno = Estrutura **PortfolioResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura PortfolioResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com dados do portfólio (pode retornar sem nenhum registro):

⬝ Symbol = Código do símbolo (ticker da bolsa):

- SourceID (string) = Id do mercado OMS (exemplo: 12010 para BVSP.VIS);

- SymbolID (string) = Código do símbolo no OMS.

⬝ Description (string) = Descrição do símbolo;

⬝ Market (string) = Mercado do símbolo:

- BMF.FUT = Segmento BMF Futuro;

- BMF.OPD = Segmento BMF Opção sobre Disponível;

- BMF.OPF = Segmento BMF Opção sobre Futuro;

- BMF.TMO = Segmento BMF Termo;

- BMF.DIS = Segmento BMF Disponível;

- BVSP.VIS = Segmento Bovespa Vista;

- BVSP.OPC = Segmento Bovespa Opção;

- BVSP.TRM = Segmento Bovespa Termo.

⬝ MarketSegment (string) = Segmento do mercado do símbolo:

- O = Opção;

- U = Futuro;

- V = Ação.

⬝ Liquidity (bool) = Se símbolo líquido True=Sim ou False=Não;

⬝ QuotesShares (string) = Fator de cotação;

⬝ TradeShare (string) = Lote de negociação;

⬝ AveragePrice (string) = Preço médio (calculado pelo back-office);

⬝ TotalQtty (string) = Quantidade total;

⬝ PendingQtty (string) = Quantidade pendente;

⬝ ConfirmedQtty (string) = Quantidade confirmada (negócio);

⬝ BloquedQtty (string) = Quantidade bloqueada;

⬝ AvailableQtty (string) = Quantidade disponível;

⬝ DayTradeQtty (string) = Quantidade confirmada (negociada) no Day-Trade;

⬝ QttyBidP (string) = Quantidade de compra pendente;

⬝ QttyAskP (string) = Quantidade de venda pendente;

⬝ QttyBidC (string) = Quantidade de compra confirmada (negócio);

⬝ QttyAskC (string) = Quantidade de venda conformada (negócio);

⬝ ValueBidC (string) = Valor de compra confirmada (negócio);

⬝ ValueAskC (string) = Valor de venda confirmada (negócio);

⬝ QttyBidTPE (string) = Quantidade de compra com situação TPE (aguardando disparo, pendente ou processada);

⬝ QttyAskTPE (string) = Quantidade de venda com situação TPE (aguardando disparo, pendente ou processada);

⬝ ValueBidTPE (string) = Valor de compra com situação TPE (aguardando disparo, pendente ou processada);

⬝ ValueAskTPE (string) = Valor de venda com situação TPE (aguardando disparo, pendente ou processada);

⬝ QttyBidXV (string) = Quantidade de compra com situação XV (cancelada ou vencida);

⬝ QttyAskXV (string) = Quantidade de venda com situação XV (cancelada ou vencida);

⬝ ValueBidXV (string) = Valor de compra com situação XV (cancelada ou vencida);

⬝ ValueAskXV (string) = Valor de venda com situação XV (cancelada ou vencida);

⬝ QttyRent (string) = Quantidade alugada.

▪ Exemplo assíncrono para solicitação da custódia do cliente ID-OMS 68975:

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Trading.Portfolio;

...

// Filtro para buscar custódia do cliente ID OMS = 68975

PortfolioParam portfolioParam = new PortfolioParam();

portfolioParam.UserId.User = 68975;

portfolioParam.PortfolioType = 2;

// Solicitar portfólio para o filtro desejado

cmaStreamerFeed.Portfolio(portfolioParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<List<PortfolioResult>> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nPortfólio consultado: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nPortfólio consultado retornou o status: " + result.Status.ToJSON());

}

}

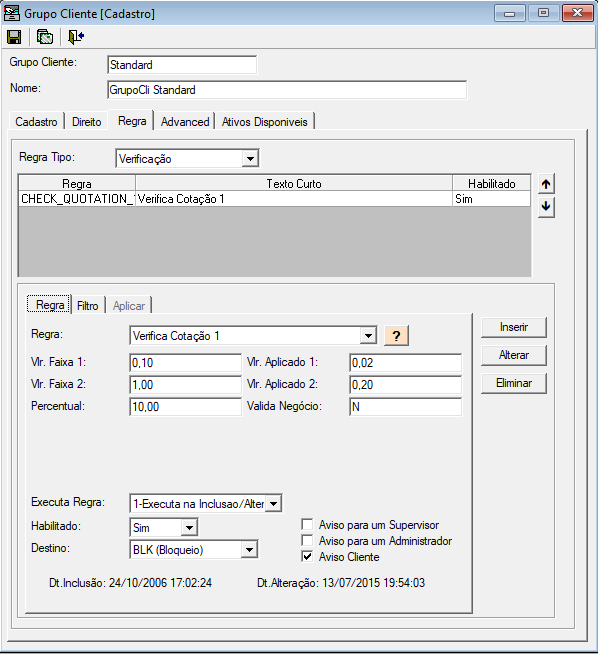
# 2.5.7) Solicitação das Regras Verifica Cotação 1/2 para API (RuleVerifyQuotes)

▪ Método **RuleVerifyQuotes** com parâmetro **RuleVerifyQuotesParam**;

▪ Serão pesquisadas as regras “*Verifica Cotação 1*” (utilizada no pregão normal) e “*Verifica Cotação 2*” (utilizada no pregão after-market) configuradas no “*Cliente*” ou no “*Grupo de Cliente*” (quando configurado para seguir o grupo);

▪ Com base nas regras “*Verifica Cotação 1/2*”, juntamente com a cotação da ação no momento da solicitação, serão calculados os preços mínimo e máximo, ou seja, o túnel de preço permitido pelas regras;

▪ Tela exemplo do AC para configuração da regra “*Verifica Cotação 1*” cadastrada no Grupo do Cliente:



Observações:

- Esta mesma regra pode ser cadastrada diretamente no Cliente;

- Informações detalhadas da regra no documento do OMS “*CMAeBroker\_Regras.doc*”.

▪ Aplicação da regra:

⇨ Considerando *ValueQuote* o valor da cotação, temos:

⬝ Se *ValueQuote* <= *ValueBand1*, aplica-se 1ª faixa:

- *ValueMin* = *ValueQuote* – *ValueApplied1*

- *ValueMax* = *ValueQuote* + *ValueApplied1*

⬝ Se *ValueQuote* <= *ValueBand2*, aplica-se 2ª faixa:

- *ValueMin* = *ValueQuote* – *ValueApplied2*

- *ValueMax* = *ValueQuote* + *ValueApplied2*

⬝ Caso contrário, aplica-se *Percentage*:

- *ValueMin* = *ValueQuote* – (*ValueQuote* \* (*Percentage*/100))

- *ValueMax* = *ValueQuote* + (*ValueQuote* \* (*Percentage*/100))

⇨ Validações considerando o *PriceOffer* o preço da ordem:

⬝ Se *UseSideOffer* = true:

- Se operação de Venda e *PriceOffer* < *ValueMin* 🡪 Erro;

- Se operação de Compra e *PriceOffer* > *ValueMax* 🡪 Erro.

⬝ Se UseSideOffer = false:

- Se *PriceOffer* < *ValueMin* 🡪 Erro;

- Se *PriceOffer* > *ValueMax* 🡪 Erro.

⇨ Mercados e Integrações:

⬝ Se não enviado indica que não tem filtro, ou seja, a regra deve ser aplicada em todas as operações;

⬝ Se enviado indica que tem filtro, ou seja, a regra deve ser aplicada apenas nos mercados/integrações enviados.

▪ Parâmetros de entrada = Estrutura **RuleVerifyQuotesParam**:

⇨ UserId = Identificação do cliente para a posição:

⬝ User [obrigatório] (long) = Id numérico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Id*”);

⬝ UserMne [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⬝ Branch [opcional] (string) = Mnemônico da filial do cliente (obtido da estrutura “*ClientList*”).

⇨ SourceID [opcional] (string) = Id da Origem do símbolo (exemplo: "12" para Bovespa);

⇨ SymbolID [obrigatório] (string) = Símbolo (exemplo: PETR4);

⇨ Side [obrigatório] (string) = B-Compra ou S-Venda;

⇨ AfterMarket [obrigatório] (bool) = Verdadeiro indica que a operação será incluída no pregão after-market.

Observação: Se enviado “*UserMne*” (mnemônico do OMS) o campo “*User*” pode ser enviado com o valor 0 (zero).

▪ Retorno = Estrutura **RuleVerifyQuotesResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura RuleVerifyQuotesResult.

⇨ Data = Dados com cálculo do túnel de preço baseado nas regras e a cotação da ação no momento da solicitação (último ou fechamento anterior):

⬝ PrevClosePrice (double) = Preço do fechamento anterior;

⬝ LastTradePrice (double) = Preço do último negócio;

⬝ LastTradeTime (string) = Horário do último negócio no formato HH:MM:SS;

⬝ PriceMinimum (double) = Preço mínimo com base nas regras "*Verifica Cotação*";

⬝ PriceMaximum (double) = Preço máximo com base nas regras "*Verifica Cotação*".

Observação: Se nenhuma regra atribuída ao cliente será retornado PriceMinimum = 0

e PriceMaximum = 9999999999.

▪ Exemplo assíncrono para solicitação da regra do cliente ID-OMS 68975:

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Trading.RuleVerifyQuotes;

...

// Filtro para buscar as regras de verificação de preço do cliente ID OMS = 68975

RuleVerifyQuotesParam RuleVerifyQuotesParam = new RuleVerifyQuotesParam();

RuleVerifyQuotesParam.UserId.User = 68975;

RuleVerifyQuotesParam.SourceID = “12”;

RuleVerifyQuotesParam.SymbolID = “PETR4”;

RuleVerifyQuotesParam.Side = “B”;

RuleVerifyQuotesParam.AfterMarket = false;

// Solicitar as regras de verificação de preço para o filtro desejado

cmaStreamerFeed.RuleVerifyQuotes (RuleVerifyQuotesParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<List<RuleVerifyQuotesResult>> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nRegras OMS: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nConsulta de regras retornou o status: " + result.Status.ToJSON());

}

}

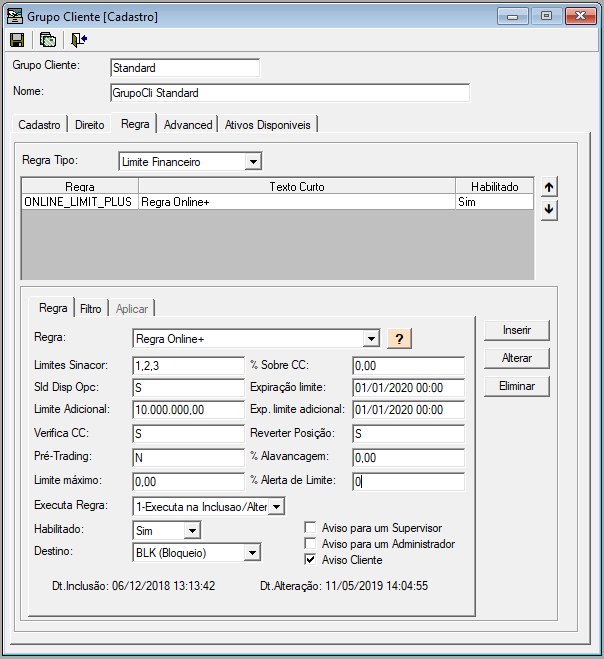
# 2.5.8) Solicitação do Extrato Financeiro (FinancialStatements)

▪ Método **FinancialStatements** com parâmetro **FinancialStatementsParam**; (liberado a partir da versão 1.0.0.356 da CMA.StreamerFeed.dll);

▪ Este método retorna informações da execução da regra “*Regra OnLine+*” (Limite OnLine+), informações do Sinacor e informações do próprio OMS;

▪ Também retorna limites financeiros calculados com as informações descritas acima;

▪ Tela exemplo do AC para configuração da regra “*Regra OnLine+*” cadastrada no Grupo do Cliente:



Observações:

- Esta mesma regra pode ser cadastrada diretamente no Cliente;

- Informações detalhadas da regra no documento do OMS “*CMAeBroker\_Regras.doc*”.

▪ Parâmetros de entrada = Estrutura **FinancialStatementsParam**:

⇨ UserId = Identificação do cliente para a posição:

⬝ User [obrigatório] (long) = Id numérico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Id*”);

⬝ UserMne [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⬝ Branch [opcional] (string) = Mnemônico da filial do cliente (obtido da estrutura “*ClientList*”).

Observação: Se enviado “*UserMne*” (mnemônico do OMS) o campo “*User*” pode ser enviado com o valor 0 (zero).

▪ Retorno = Estrutura **FinancialStatementsResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura FinancialStatementsResult.

⇨ Data = Dados do Extrato Financeiro:

**⭢ Informações do Sinacor (view vwCMAEBSALDOSCC)**

⬝ DtD0 (string) = Data Atual (formato YYYY-MM-DD);

⬝ DtD1 (string) = Data Atual + 1 (formato YYYY-MM-DD);

⬝ DtD2 (string) = Data Atual + 2 (formato YYYY-MM-DD);

⬝ DtD3 (string) = Data Atual + 3 (formato YYYY-MM-DD);

⬝ VlDisponivel (string) = Valor Disponível;

⬝ VlTotal (string) = Valor Total;

⬝ VlProjetado (string) = Valor Projetado;

⬝ VlD1 (string) = Valor Data Atual + 1;

⬝ VlD2 (string) = Valor Data Atual + 2;

⬝ VlD3 (string) = Valor Data Atual + 3;

⬝ VlOperacional1 (string) = Valor Limite Operacional 1;

⬝ VlOperacional2 (string) = Valor Limite Operacional 2;

⬝ VlOperacional3 (string) = Valor Limite Operacional 2;

**⭢ Informações do Sinacor (views vwCMAEBSALDOSCC e vwCMAEBEXTRATO)**

⬝ SaldoInicialEx (string) = Saldo Inicial do Extrato;

⬝ MovimentacaoDiaEx (string) = Movimentação na Data Atual do Extrato;

⬝ LiquidacaoDiaEx (string) = Liquidação na Data Atual do Extrato;

**⭢ Informações da Posição no OMS (tabela POSITION\_CLIENT)**

⬝ VisValueBidC (string) = Valor Compra Bovespa Vista Confirmada;

⬝ VisValueBidTpe (string) = Valor Compra Bovespa Vista Pendente;

⬝ VisValueAskC (string) = Valor Venda Bovespa Vista Confirmada;

⬝ VisValueAskTpe (string) = Valor Venda Bovespa Vista Pendente;

⬝ OpcValueBidC (string) = Valor Compra Bovespa Opção Confirmada;

⬝ OpcValueBidTpe (string) = Valor Compra Bovespa Opção Pendente;

⬝ OpcValueAskC (string) = Valor Venda Bovespa Opção Confirmada;

⬝ OpcValueAskTpe (string) = Valor Venda Bovespa Opção Pendente;

**⭢ Informações Financeiras no OMS (tabela CASH\_CLIENT)**

⬝ ValueLiquidationVis (string) = Valor Liquidação Mercado Vista;

⬝ ValueLiquidationOpc (string) = Valor Liquidação Mercado Opção;

**⭢ Informações Calculadas na Regra OnLine+ (regra ONLINE\_LIMIT\_PLUS)**

⬝ VlDisponivelD1D2D3 (string) = Valor Disponível + Valores Opercionais Sinacor;

⬝ VlDisponivelSobreCC (string) = Valor Sobre % Disponível C/C;

⬝ VisValue (string) = Valor Operado no dia Bovespa Vista (vendas - compras);

⬝ OpcValue (string) = Valor Operado no dia Bovespa Opção (vendas - compras);

⬝ DailyLimit (string) = Limite no Dia (vlr.alav + vlr.dispcc + vlr.oper1/2/3);

⬝ VlDisponivelVis (string) = Valor Disponível Bovespa Vista;

⬝ VlDisponivelOpc (string) = Valor Disponível Bovespa Opção;

⬝ VlDisponivelRes (string) = Valor Disponível para Resgate;

⬝ FinancialLimit (string) = Limite Financeiro Cliente;

⬝ PercSobreCC (string) = Regra OnLine+ %Sobre C/C;

⬝ Leverage (string) = % Alavancagem Cliente;

⬝ ValueLeverage (string) = Valor Alavancagem Cliente;

⬝ VlDisponivelFinal (string) = Valor Disponível (limites e % C/C);

⬝ ValueLeverageInitial (string) = Valor Alavancagem da Posição Bovespa Vista;

**⭢ Informações de Corretagem calculadas (corretagem configurada no OMS)**

⬝ ValueBrokerage (string) = Valor da Corretagem;

⬝ ValueBrokerageFee (string) = Valor Emolumentos e Taxas;

⬝ ValueBrokerageDevolution (string) = Valor Devolução de Corretagem;

**⭢ Informações Calculadas no FinancialStatements**

⬝ LimitInitial (string) = Valor Limite Financeiro Inicial;

⬝ LimitCurrent (string) = Valor Limite Financeiro Atual;

⬝ LimitFinal (string) = Valor Limite Final;

⬝ LimitCredit (string) = Valor Limite Crédito.

▪ Exemplo assíncrono para solicitação da regra do cliente ID-OMS 68975:

...

using System;

using System.Collections.Generic;

using CMA.StreamerFeed.Models.Interfaces;

using CMA.StreamerFeed.Models.V1;

using CMA.StreamerFeed.Models.V1.Trading.FinancialStatements;

...

// Filtro para buscar o extrato do cliente ID OMS = 68975

FinancialStatementsParam FinancialStatementsParam = new FinancialStatementsParam();

FinancialStatementsParam.UserId.User = 68975;

// Solicitar o extrato para o filtro desejado

cmaStreamerFeed. FinancialStatements(FinancialStatementsParam, 10000).Subscribe(

async obs =>

{

try

{

AsyncResultTestWriter(await obs);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

},

ex => Console.WriteLine(ex.Message)

).Dispose();

...

static void AsyncResultTestWriter(Result<FinancialStatementsResult> result)

{

if (result.Status.Success)

{

Console.WriteLine($"\nExtrato Financeiro OMS: " + result.ToJSON());

}

else

{

Console.WriteLine($"\nConsulta do extrato financeiro retornou o status: " + result.Status.ToJSON());

}

}

# 2.5.9) Solicitação de Market Symbol Info para a API

▪ Método **MarketSymbolInfo**;

▪ Parâmetros de entrada = Estrutura **MarketSymbolInfoParam**:

⇨ SourceID [obrigatório] (string) = Id da Origem do símbolo (exemplo: “12” para Bovespa);

⇨ Symbols [obrigatório] (list string) = Símbolos (exemplo: PETR4);

▪ Retorno = Estrutura **MarketSymbolInfoResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura MarketSymbolInfoResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Coleção com a lista de símbolos (pode retornar sem nenhum símbolo):

⬝ SourceID (string) = Id da Origem no Market Data;

⬝ Símbolo (string) = Símbolo;

⬝ MarketId (integer) = Id do Mercado no OMS;

⬝ MinimumShare (string) = Lote de Negociação;

⬝ PriceFactor (string) = Fator de Cotação;

⬝ FactorValue (string) = Fator de Cálculo;

⬝ CurrentMarketSession = Sessão Atual de mercado:

⬝ Found (boolean) = Indicador se localizado período;

⬝ PeriodId (string) = Id do Período do OMS;

⬝ Date (string) = Data no formato YYYY-MM-DD;

⬝ StartTime (string) = Início do Mercado (em número de minutos);

⬝ EndTime (string) = Final do Mercado (em número de minutos);

⬝ After (boolean) = Indicador se tem after-market.

⬝ NextMarketSession = Próxima Sessão de mercado:

⬝ Found (boolean) = Indicador se localizado período;

⬝ PeriodId (string) = Id do Período do OMS;

⬝ Date (string) = Data no formato YYYY-MM-DD;

⬝ StartTime (string) = Início do Mercado (em número de minutos);

⬝ EndTime (string) = Final do Mercado (em número de minutos);

⬝ After (boolean) = Indicador se tem after-market.

⬝ Message (string) = Informações sobre o símbolo.

# 2.5.10) Solicitação de Lista de Mercado e Símbolo para a API

▪ Método **ListMarketSymbol**;

▪ Parâmetros de entrada = Estrutura **ListMarketSymbolParam**:

⇨ SymbolFilter [obrigatório] (list string) = Filtro de Símbolos (exemplo: PETR);

⇨ DaytradeEnabled [opcional] (bool) = Se true retorna apenas os símbolos que permitem Day-Trade (default false);

⇨ PageSize [opcional] (integer) = Número de símbolos em cada página;

⇨ SubscriptionId [opcional] (string) = Id da Subscrição;

⇨ PageId [opcional] (integer) = Id da Página que deve retornar.

▪ Retorno = Estrutura **ListMarketSymbolResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura MarketSymbolInfoResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Resultado paginado com dos símbolos:

⬝ Symbols = Lista de Símbolos:

⬝ MarketId (integer) = Id do Mercado (exemplo: 12010 para Bovespa Vista);

⬝ Symbol (string) = Ticker do Símbolo;

⬝ Description (string) = Descrição do Símbolo;

⬝ DaytradeEnabled (string) = True-DayTrade ou False-Não DayTrade;

⬝ Source (integer) = Origem CMA.

⬝SubscriptionId string (bool) = Id da Subiscrição;

⬝PageQuantity int (integer) = Quantidade de Páginas;

⬝PageSize int (integer) = Tamanho da Página;

⬝PageId int (integer) = Número da Página.

# 2.5.11) Solicitação de Exercício de Opção para a API

▪ Método **SendOptionExercise**;

▪ Parâmetros de entrada = Estrutura **SendOptionExerciseParam**:

⇨ UserId = Identificação do cliente para o qual a ordem será atribuída:

⬝ User [obrigatório] (long) = Id numérico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Id*”);

⬝ UserMne [opcional] (string) = Mnemônico do cliente (obtido da estrutura “*ClientListResult*” campo “*Data.Mnemonic*”);

⬝ Branch [opcional] (string) = Mnemônico da filial do cliente (obtido da estrutura “*ClientListResult*”).

⇨ SymbolId = Código do símbolo (opção para exercício):

⬝ SourceID (string) = Id da Origem na CMA (exemplo: 12 para Bovespa);

⬝ Symbol (string) = Código do símbolo no OMS.

⇨ MarketId = Código do mercado OMS:

⬝ Market (string) = Id do mercado OMS (exemplo: 12020 para BVSP.OPC);

⬝ MarketMne (string) = Mnemônico do mercado OMS.

⇨ Qtty [obrigatório] (string) = Quantidade da ordem;

⇨ Confirmation [obrigatório] (bool) = Indicativo se o envio da ordem está sendo utilizado para montar a tela de confirmação (True - default) ou se a ordem deve ser gravada no OMS (False);

⇨ IpRemote [opcional] (string) = Endereço ip do requisitante;

⇨ HostName [opcional] (string) = Nome do host do requisitante;

⇨ Signature [opcional] (string) = Assinatura eletrônica (se configurado no Streamer).

▪ Retorno = Estrutura **SendOptionExerciseResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura MarketSymbolInfoResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Resultado da solicitação de exercício:

⬝ ExerciseId (string) = Id do Exercício (ordem de exercício esta;

⬝ AddDate (string) = Data da inclusão no formato YYYY-MM-DD HH:MM:SS;

⬝ TypeSuitability (string) = Tipo do termo Suitability a ser mostrado ao cliente;

⬝ AlertText (string) = Mensagem de alerta Suitability;

⬝ AgreementText (string) = Mensagem para o termo de aceite;

⬝ Profile (string) = Perfil do investidor;

⬝ NewProfile (string) = Novo perfil do investidor considerando esta nova ordem;

⬝ Textual (string) = Texto gerado pelo OMS para confirmação da ordem;

⬝ Blocked (bool) = True indica bloqueado.

# 2.5) Solicitação Diversas para API

# 2.5.1) Solicitação de Load Test para a API

▪ Método **LoadTest**;

LoadTest(LoadTestParam loadTestParam, CancellationToken cancelToken, int millisecondsToRequestLooping = 300, int millisecondsTimeOut = 10000)

Onde:

- LoadTestParam = Parâmetros do teste (vide abaixo);

- CancellationToken = Token que permite cancelar a assinatura (gerar token com o comando “*new CancellationTokenSource()*”);

- millisecondsToRequestLooping = Tempo (em milissegundos) para execução do loop (default 300ms);

- millisecondsTimeOut = Tempo (em milissegundos) para timeout do teste (default 10000ms).

*Atenção*: Utilizar o token gerado para efetuar o cancelamento da assinatura (comando “*cancellationTokenSource.Cancel();*”) que não está sendo mais utilizada, permitindo assim reduzir o consumo de memória, processador e rede.

▪ Parâmetros de entrada = Estrutura **LoadTestParam**:

⇨ QttActors [obrigatório] (int) = Quantidade de atores que devem executar o teste simultaneamente;

⇨ QuotesParam [obrigatório] = Vide estrutura QuotesParam no item 2.4.1;

⇨ OrderSingleParam [obrigatório] = Vide estrutura OrderSingleParam no item 2.5.2;

⇨ Sign [obrigatório] (bool) = Assinatura para execução (se true o campo millisecondsToRequestLooping do método será usado como intervalo para entre as execuções do teste).

▪ Retorno = Estrutura **LoadTestResult**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura MarketSymbolInfoResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.;

⇨ Data = Resultado do teste.

▪ Vide CMA.StreamerFeedAPI.WinForm.Testes com exemplo (item 4.1).

# 2.5.2) Solicitação de Provider ⭢ DBFeed para a API

▪ Método **ProviderControl**;

ProviderControll(ProviderParam providerParam, CancellationToken cancellationToken)

Onde:

- ProviderParam = Parâmetros do DBFeed (vide abaixo);

- CancellationToken = Token que permite cancelar a assinatura (gerar token com o comando “*new CancellationTokenSource()*”).

*Atenção*: Utilizar o token gerado para efetuar o cancelamento da assinatura (comando “*cancellationTokenSource.Cancel();*”) que não está sendo mais utilizada, permitindo assim reduzir o consumo de memória, processador e rede.

▪ Parâmetros de entrada = Estrutura **ProviderParam**:

⇨ ProviderType [obrigatório] (string) = Tipo de Banco de Dados com os valores 1-MS/SQL Server, 2-MySQL e 3-Oracle);

⇨ ConnectionString [obrigatório] (string) = String de conexão para o banco de dados MySQL, MS/SQL ou Oracle;

⇨ QuotesParam [obrigatório] (lista de QuotesParam) = Vide estrutura QuotesParam no item 2.4.1;

⇨ QuotesTemplate [obrigatório] (string) = Template (Fluid engine de template baseado na linguagem de template Liquid) para salvar na base de dados as cotações recebidas.

▪ Retorno = Estrutura **StatusProvider**:

⇨ Status:

⬝ Success (bool) = True ou False;

⬝ Code (string) = Código do erro (se success=False);

⬝ Message (string) = Mensagem do erro (se success=False);

⬝ OmsCode (string) = Código do erro OMS (se success=False);

⬝ StreamerCode (string) = Código do erro Streamer (se success=False);

⬝ Referer (object) = Objeto para identificar a requisição no momento da resposta (responsabilidade do serviço);

⬝ ToJSON (string) = Conversor para JSON da estrutura MarketSymbolInfoResult.

⇨ EnableTrace (bool) = Vide item 2.3.1.;

⇨ Trace = Coleção da estrutura TraceItems = Vide item 2.3.1.

▪ Vide CMA.StreamerFeedAPI.WinForm.Testes com exemplos (item 4.2).

# 3) Informações Técnicas do CMA Streamer Feed REST

▪ O REST permite dois cenários de instalação, são eles:

1º Site do MS/IIS (Microsoft Internet Information Service)

2º Diretamente no EXE utilizando Kestrel Web Server (interno ao aplicativo)

▪ Compilado em .NET 5.0 para Plataforma X64 (utiliza Streamer Feed API .NET Standard 2.0);

▪ Funções Market Data disponibilizadas no REST:

⇨ **Quotes =** Solicitar cotações de uma lista de símbolos;

⇨ **Symbol =** Solicitar lista de símbolos de acordo com filtro;

⇨ **Book =** Solicitar livro de ofertas de um símbolo;

⇨ **HistoricalData =** Solicitar pontos do gráfico de um símbolo;

⇨ **Highlights =** Solicitar relatórios;

⇨ **News =** Solicitar manchete das notícias;

⇨ **News/Body =** Solicitar íntegra da notícia;

▪ Funções Negociação Renda Variável disponibilizadas no REST:

⇨ **OrderSingle =** Inclusão de Ordem;

⇨ **Orderreplace =** Alteração de Ordem;

⇨ **OrderCancel =** Cancelamento de Ordem;

⇨ **OrderList =** Lista de Ordens;

⇨ **Porffolio =** Custódia.

**Atenção:**

- O REST utiliza os mesmos parâmetros de entrada e saída da API documentados nos itens “*2.4) Solicitação de Informações de Market Data para API*” e “*2.5) Solicitação de Informações e Transações de Negociação para API*”;

- No item “*3.3) Testando o REST via Postman*” é possível obter exemplo para todas as funções citadas acima.

# 3.1) Informações, Configurações e Instalação

# 3.1.1) Versionamento e Distribuição

▪ Versionamento no padrão A.B.C.F “*major.minor.branch.fix-release*”:

⇨ A - major = Grandes modificações e mudanças e quebra de compatibilidade;

⇨ B - minor = Novas funcionalidades mantendo compatibilidade;

⇨ C - branch = Ramificação com novas funcionalidades e ajustes mantendo compatibilidade;

⇨ F - fix-release = Correções de falhas e ajustes mantendo compatibilidade.

▪ Distribuição através de pacote completo com EXE, DLL (API) e Dependências.

# 3.1.2) Configurações

▪ Configurações do REST estão no arquivo “*appsettings.json*” (lido no momento do startup), segue exemplo:

{

"JWT": {

"Secret": "D5A49C08F6383A5BA97E4158EA34631BA69D968FDEB15FF738ADCA1F73A8D8DA",

"HoursToExpire": 24

},

"ConnectionStrings": {

"DefaultConnectionMySQL": "Server=127.0.0.1;Port=3306;Database=CMAStreamerFeed;Uid=dev;Pwd=123;"

},

"Cache": {

"Template": {

"SlidingExpirationMinutes": 1

}

},

"IPAddressFilterConfiguration": {

"AuthorizedIPAddresses": ["\*"]

}

}

🢞 Detalhes:

⇨ JWT - Secret (obrigatório): Chave Secreta para criptografia do Token do REST (recomendamos no mínimo 64 caracteres);

⇨ JWT - HoursToExpire (opcional): Passados as horas configuradas é necessário chamar novamente “*Authenticate*”, caso contrário as requisições serão recusadas com o Status 401 (Unauthorized);

⇨ Cache - Template - SlidingExpirationMinutes (obrigatório): Minutos para expiração do cache de templates;

⇨ IPAddressFilterConfiguration - AuthorizedIPAddresses (obrigatório): Hosts autorizados a utilizar o REST, exemplo:

"IPAddressFilterConfiguration": {

"AuthorizedIPAddresses": [

"::1", // IPv6 localhost

"127.0.0.1", // IPv4 localhost

"192.168.0.0/16", // IPv4 local network CIDR 16 Mask Bits

"10.0.0.0/24", // IPv4 local network CIDR 24 Mask Bits

"10.10.100.73" // IPv4 local network

]

}

Observações:

- Permite utilizar CIDR <https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing>;

- Site para calcular CIDR: <https://mxtoolbox.com/subnetcalculator.aspx>.

🢞 Quando utilizado banco de dados MySQL configurar a conexão conforme exemplo:

"DefaultConnectionMySQL": "Server=127.0.0.1;Port=3306;Database=CMAStreamerFeed;Uid=dev;Pwd=123;"

🢞 Quando utilizado banco de dados MS/SQL configurar a conexão conforme exemplo:

"DefaultConnectionSQLServer": "Data Source=127.0.0.1;Initial Catalog=CMAStreamerFeed;User ID=dev;Password=123"

Observação: A base de dados e estrutura são criadas automaticamente durante a primeira execução, porém o script para criação da base de dados e estrutura para MySQL pode ser encontrada no arquivo “*CMA\_Streamer\_Feed\_REST-MySQL-Script.sql*”.

▪ Configurações da API estão no arquivo “*CMA.StreamerFeed.REST.dll.config*”, segue exemplo:

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<appSettings>

<!--

Níveis de configuração do Serilog:

1-Verbose - informações de rastreamento e minúcias de depuração; geralmente apenas ligado em situações incomuns

2-Debug - fluxo de controle interno e dumps de estado de diagnóstico para facilitar a identificação de problemas reconhecidos

3-Information - eventos de interesse ou que tenham relevância para observadores externos; o nível de registro mínimo ativado padrão

4-Warning - indicadores de possíveis problemas ou degradação de serviço / funcionalidade

5-Error - indicando uma falha no aplicativo ou no sistema conectado

6-Fatal - erros críticos que causam falha completa do aplicativo-->

<add key="CMA.StreamerFeed.Api:serilog:minimum-level" value="Verbose" />

<add key="CMA.StreamerFeed.Api:serilog:using:File" value="Serilog.Sinks.File" />

<add key="CMA.StreamerFeed.Api:serilog:write-to:File.path" value=".\log\CMA.StreamerFeed.log" />

<add key="CMA.StreamerFeed.Api:serilog:write-to:File.rollingInterval" value="Day" />

<add key="CMA.StreamerFeed.Api:serilog:write-to:File.shared" value="true" />

<add key="CMA.StreamerFeed.Api:proxy:url" value="" />

<!--Ex.: "http://myproxyserver"-->

<add key="CMA.StreamerFeed.Api:proxy:user" value="" />

<add key="CMA.StreamerFeed.Api:proxy:password" value="" />

<!--Quantidade Máxima de objetos no Cache-->

<add key="CMA.StreamerFeed.Api:cache-manager:SizeLimit" value="1000" />

<!--Quantidade de horas para um objeto expirar no cache-->

<add key="CMA.StreamerFeed.Api:cache-manager:AbsoluteExpirationHours" value="24" />

<!--Quantidade de horas para um login expirar no cache-->

<add key="CMA.StreamerFeed.Api:acceptor-logins:SlidingExpirationHours" value="1" />

</appSettings>

</configuration>

Observação: O REST vai salvar os logs no mesmo arquivo de log da API.

# 3.1.3) Authenticate e Token

▪ Antes de iniciar as solicitações via REST é necessário efetuar autenticação, segue exemplo:

<http://localhost:8083/Authenticate>

▪ No momento da autenticação é necessário enviar no corpo do método POST o JSON:

{

"InstanceName": "TesteCMA-REST",

"AcceptorUrl": "http://devwebfrontendstr.cma.com.br:9080/acceptor",

"DomainId": "streamerfeedapi",

"User": "xyz",

"Password": "123",

"Service": "m",

"CustomerId": "",

"IpRemote": "127.0.0.1",

"HostName": "cma.local",

"LoginTokenSecretKey" value="",

"LoginTokenIvKey" value="",

"LoginTokenSaltKey" value="",

"LoginTokenSaltIterations" value="65536",

"LoginTokenKeyLength" value="128"

}

🢞 Onde:

⇨ InstanceName (obrigatório): Nome da Instância;

⇨ AcceptorUrl (obrigatório): URL(s) do CMA Web Acceptor (via “*load balance*” ou lista de Acceptors);

⇨ DomainId (obrigatório): Domain de Streamers;

⇨ User e Password (obrigatório): Mnemônico e Senha do usuário Market Data;

⇨ Service (obrigatório): Tipo de serviço utilizado “m” (MarketData) ou “n” (Negociação via OMS);

⇨ CustomerId (opcional): Identificação única do usuário (se não enviado será utilizado “*hostname*” + GUID);

⇨ IpRemote (opcional): Endereço ip do requisitante;

⇨ HostName (opcional): Nome do host do requisitante;

⇨ LoginTokenSecretKey (opcional): AES Secret-Key (se informado indica que será utilizado Token);

⇨ LoginTokenIvKey (opcional): ES IV-Key (obrigatório se informado LoginTokenSecretKey);

⇨ LoginTokenSaltKey (opcional): AES Salt-Key (obrigatório se informado LoginTokenSecretKey);

⇨ LoginTokenSaltIterations (opcional): AES Salt Iterations (obrigatório se informado LoginTokenSecretKey);

⇨ LoginTokenKeyLength (opcional): Length of Key (obrigatório se informado LoginTokenSecretKey);

🢞 Se autenticação correta deve retornar o token, segue exemplo:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..rjQScf6HsG-v7TJgv9hfGYTm7JxTZaVVQIbrMxBJTbQ

▪ Nas demais requisições (Quotes, News, etc) o token recebido na autenticação deve ser enviado no modo Bearer Token;

▪ O token recebido na autenticação tem validade, por este motivo se receber o erro 401 (Unauthorized) uma nova autenticação deve ser executada para obter novo token.

# 3.1.3) Instalando CMA Streamer Feed REST no MS/IIS

▪ Instalado em um site do MS/IIS (Microsoft Internet Information Service);

▪ Necessário instalar o Gerenciador dos Serviços de Informações da Internet (IIS) no “Painel de Controle ⭢ Programas e Recursos ⭢ Ativar Recursos do Windows”;

▪ Necessário instalar ASP.NET Core 5.0 Runtime Windows Hosting Bundle, segue informações do instalador:

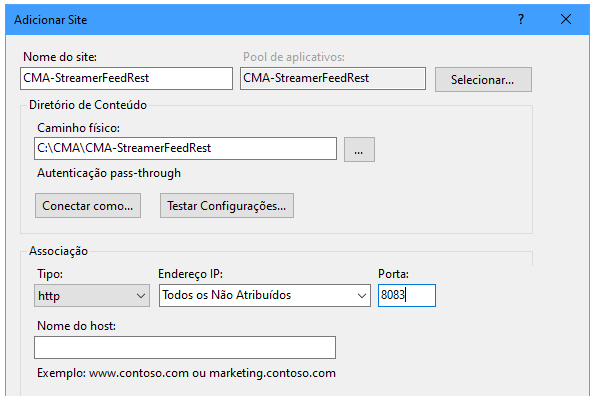
🢞 URL = <https://dotnet.microsoft.com/permalink/dotnetcore-current-windows-runtime-bundle-installer>;

🢞 Arquivo = dotnet-hosting-5.0.5-win.exe (65KB).

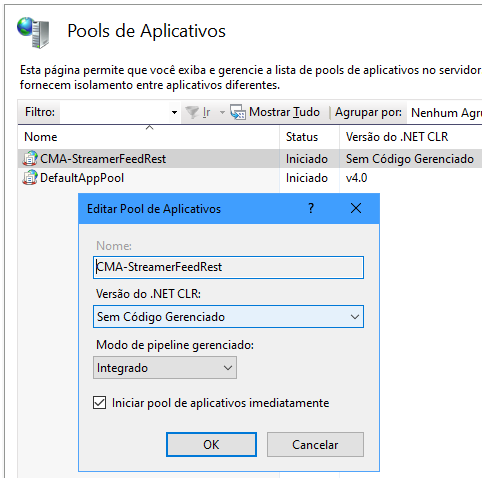
▪ Configurar Gerenciador dos Serviços de Informações da Internet (IIS):

🢞 Copiar diretório do REST para uma pasta do servidor (exemplo: C:\CMA\CMA-StreamerFeedRest);

🢞 Criar novo Site para hospedar o CMA Streamer Feed REST, exemplo “*CMA-StreamerFeedRest*”:



🢞 Configurar o Pool de Aplicativos para “*Sem Código Gerenciado*”:



🢞 Se instalado conforme exemplo acima é possível testar via Postman (anexo no documento vide item 3.3) acessando a URL <http://localhost:8083/Authenticate>.

# 3.1.4) Instalando CMA Streamer Feed REST como executável (EXE)

▪ O *CMA.StreamerFeed.REST.exe* utiliza o webserver “*Kestrel Web Server*”;

▪ Configurar portas http e https (certificado gerado pelo Kestrel): Adicionar a linha com os endereços desejados separador por “;”, segue exemplo arquivo “*appsettings.json*”:

{

"JWT": {

"Secret": "D5A49C08F6383A5BA97E4158EA34631BA69D968FDEB15FF738ADCA1F73A8D8DA",

"HoursToExpire": 24

},

"ConnectionStrings": {

"DefaultConnectionMySQL": "Server=127.0.0.1;Port=3306;Database=CMAStreamerFeed;Uid=dev;Pwd=123;"

},

"Cache": {

"Template": {

"SlidingExpirationMinutes": 1

}

},

"IPAddressFilterConfiguration": {

"AuthorizedIPAddresses": ["\*"]

},

**"Urls": "http://localhost:8083;https://localhost:8443"**

}

▪ Configurar certificado SSL próprio: Adicionar as linhas com os endereços e certificados desejados separador por “,”, segue exemplo arquivo “*appsettings.json*”:

{

"JWT": {

"Secret": "D5A49C08F6383A5BA97E4158EA34631BA69D968FDEB15FF738ADCA1F73A8D8DA",

"HoursToExpire": 24

},

"ConnectionStrings": {

"DefaultConnectionMySQL": "Server=127.0.0.1;Port=3306;Database=CMAStreamerFeed;Uid=dev;Pwd=123;"

},

"Cache": {

"Template": {

"SlidingExpirationMinutes": 1

}

},

"IPAddressFilterConfiguration": {

"AuthorizedIPAddresses": ["\*"]

},

**"Kestrel": {**

**"Endpoints": {**

**"Http": {"Url":"http://localhost:8083"},**

**"Https": {"Url": "https://localhost:8443", "Certificate": { "Path": "<arquivo>.pfx", "Password": "<senha>"}}**

**}**

**}**

}

▪ Se desejar que o REST fique em “*listen*” em todas as interfaces de rede (endereços IPs) substituir “*localhost*” por “*0.0.0.0*”;

▪ Recomendamos o uso do CMA Home Service para instanciar o EXE;

▪ Se instalado conforme configurações acima é possível testar via Postman (anexo no documento vide item 3.3) acessando a URL <http://localhost:8083/Authenticate>.

# 3.2) Informações dos Templates

O processo de geração do resultado do REST utiliza o Fluid .Net (engine de template baseado na linguagem de template Liquid).

Mais detalhes da linguagem de template Liquid em <https://shopify.github.io/liquid/>.

O REST retorna as informações nos formatos:

▪ **JSON**

- Padrão de retorno (quando não enviado);

- Exemplo de chamada: <http://localhost:8080/Quotes>;

- Não permite uso de template (retorna todos os campos no formato JSON);

- No Content-Type do Header retorna o valor “*application/json*”.

▪ **XML**

- Exemplo de chamada: <http://localhost:8080/Quotes/xml>;

- Não permite uso de template (retorna todos os campos no formato XML);

- No Content-Type do Header retorna o valor “*application/xml*”.

▪ **CSV**

- Exemplo de chamada:

<http://localhost:8080/Quotes/csv> (utiliza o template template-csv-cotacao).

- Exemplo de chamada utilizando template:

<http://localhost:8080/Quotes/csv/template-csv-cotacao>.

- No Content-Type do Header retorna o valor “*application/octet-stream*”.

▪ **HTML**

- Exemplo de chamada:

<http://localhost:8080/Quotes/html> (utiliza o template template-html-cotacao).

- Exemplo de chamada utilizando template:

<http://localhost:8080/Quotes/html/template-html-cotacao>.

- No Content-Type do Header retorna o valor “*text/plain*”.

▪ **TXT**

- Exemplo de chamada:

<http://localhost:8080/Quotes/txt> (utiliza o template template-txt-cotacao).

- Exemplo de chamada utilizando template:

<http://localhost:8080/Quotes/txt/template-txt-cotacao>.

- No Content-Type do Header retorna o valor “*text/plain*”;

- No Content-Disposition do Header o valor “*attachment; filename=<arquivo>.txt; filename\*=UTF-8''<arquivo>.txt*”.

Observação: Os templates “*template-csv-cotacao*”, “*template-html-cotacao*” e “*template-txt-cotacao*” são criados automaticamente na primeira execução do REST (vide tabelas no item 3.2.1).

3.2.1) Base de Dados com os Templates

▪ Tabela ***Domain***:

- Tabela com os “*domains*” habilitados;

- Campos:

🢞 DomainId - integer = Id do Domain;

🢞 Name - varchar(450) = Nome do Domain.

▪ Tabela ***Template***:

- Cadastro de “*templates*”;

- Campos:

🢞 TemplateId - integer = Id do Template;

🢞 Name - varchar(250) = Nome do Template (usado na chamada do REST);

🢞 Description - varchar(500) = Descrição do Template;

🢞 Content - longtext = Texto do Template.

Observação: Após inclusão na tabela “*Template*” é necessário adicionar na tabela “*TemplateDomain*” para quais domains o novo Template pode ser acessado.

▪ Tabela ***TemplateDomain***:

- Cadastro de “*domains*” vs “*templates*”;

- Campos:

🢞 TemplateId - integer;

🢞 DomainId - integer;

🢞 SecondsToRenderedCache - integer = Minutos para expiração do cache de templates renderizados.

Observação: Os templates “*template-csv-cotacao*”, “*template-html-cotacao*” e “*template-txt-cotacao*” são criados automaticamente na primeira execução do REST.

3.2.2) Exemplo de Template para Quotes

▪ **CSV**

Symbol, LastTradePrice, PrevClosePrice

{% for q in Model %}

{{q.Data.Symbol.SymbolID}},

{{q.Data.LastTradePrice}},

{{q.Data.PrevClosePrice }}

{% endfor %}

▪ **HTML**

<table>

<thead>

<tr>

<th>Symbol</th><th>LastTradePrice</th><th>PrevClosePrice</th>

</tr>

</thead>

<tbody>

{% for q in Model %}

<tr>

<td>{{q.Data.Symbol.SymbolID}}</td>

<td>{{q.Data.LastTradePrice}}</td>

<td>{{q.Data.PrevClosePrice }}</td>

</tr>

{% endfor %}

</tbody>

</table>

▪ **TXT**

Template de Quotes

{% for q in Model %}

SymbolID: {{q.Data.Symbol.SymbolID}}

LastTradePrice: {{q.Data.LastTradePrice}}

PrevClosePrice: {{q.Data.PrevClosePrice}}

{% endfor %}

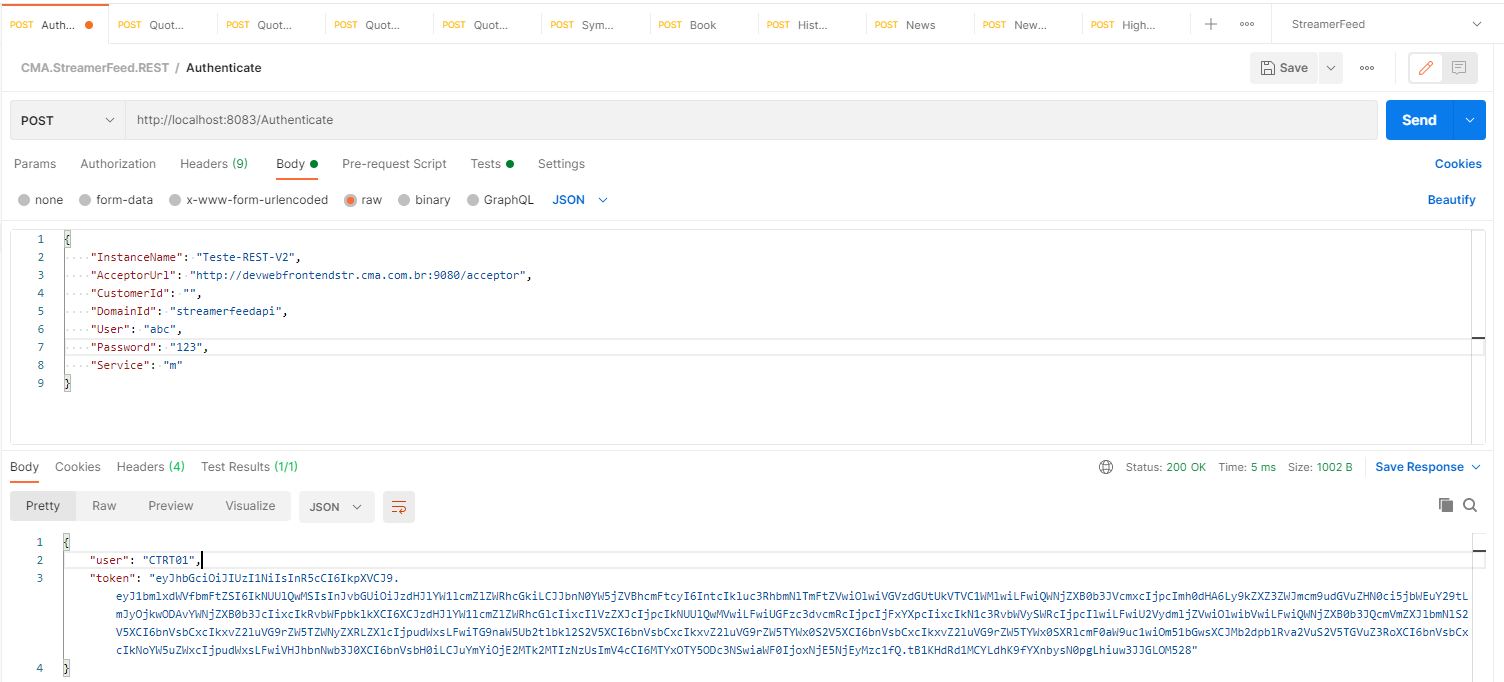
Observação: As estruturas de retorno podem ser consultadas no item “*2.4) Solicitação de Informações de Market Data para API*”.

# 3.3) Testando o REST via Postman

▪ Através do Postman (<https://www.postman.com/downloads/>) é possível acessar todos os métodos do REST;

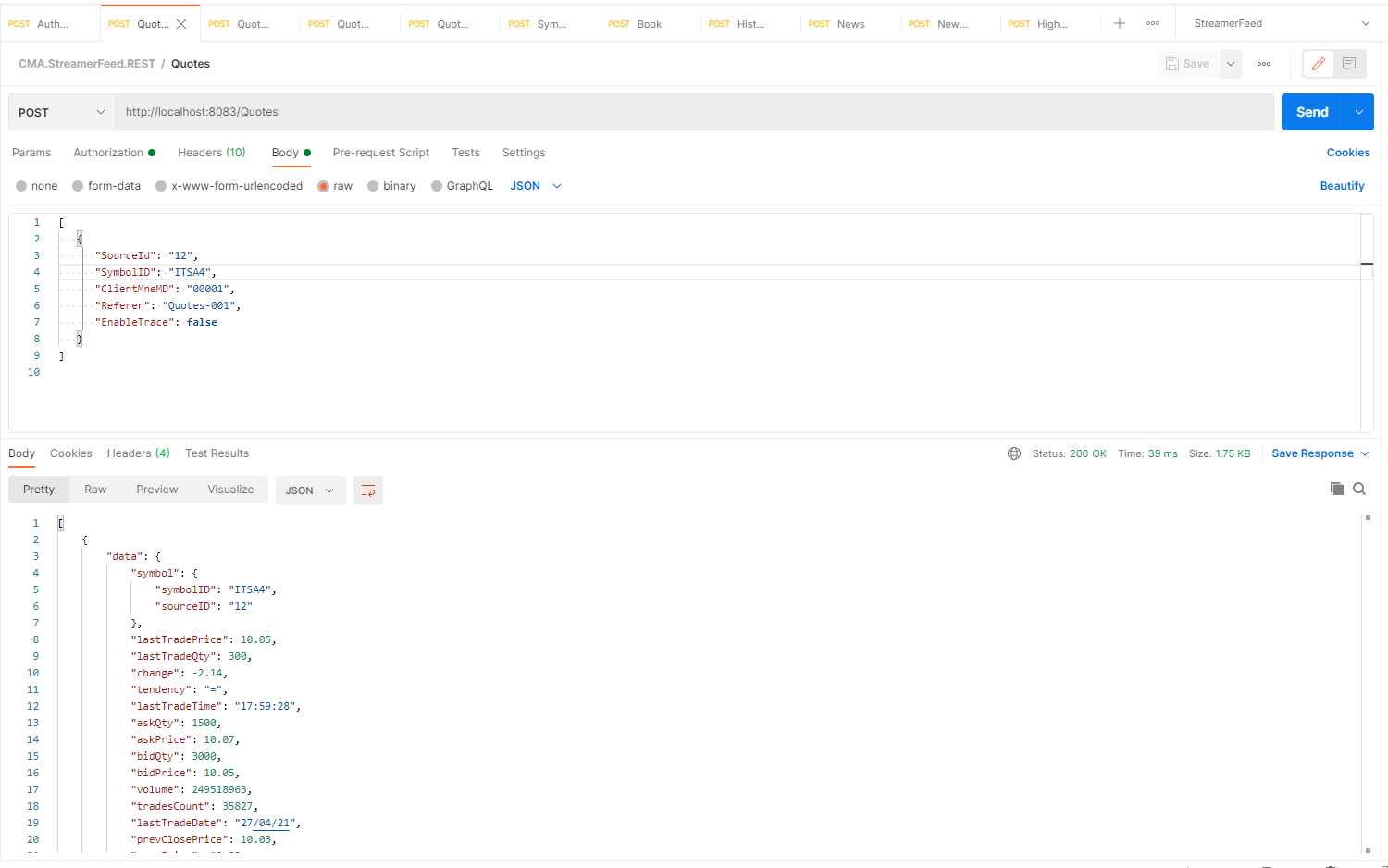
▪ Arquivo do Postman para “*Import*” com vários exemplos de chamadas liberado junto com o Pacote (pasta Documentos arquivo “*CMA.StreamerFeed.REST.postman\_collection.json*”);

▪ O método “*Authenticate*” efetua a autenticação e armazena na variável do Postman “*token*” para as próximas solicitações, segue exemplo:



Observação: Ajustar os parâmetros para login no Market Data.

▪ Após o “*Authenticate*” já podemos chamar outros métodos, segue exemplo de Quotes:



# 4) Detalhes CMA.StreamerFeedAPI.WinForm.Tester

# 4.1) Função LoadTest

▪ Exemplo de implementação para efetuar testes funcionais e carga:

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

▪ Informações iniciais para o teste:

🢞 Quantidade de Atores = Quantidade de atores simultâneos;

🢞 Ordem = Envio de Ordem e solicitação de Lista de Ordens;

🢞 Cotação = Solicitação de Cotação;

🢞 Assinar = Se assinalado o Tester vai executar o teste solicitado a cada intervalo de tempo configurado;

🢞 Looping = Intervalo de tempo (em milissegundos) para nova execução do teste quando selecionado “*Assinar*”.

▪ Informações do grid de Acompanhamento quando selecionado “*Assinar*”:

🢞 LoadTestId = Numeração sequencial dos testes executados;

🢞 QttActors = Quantidade de atores do teste;

🢞 Looping (ms) = Intervalo de tempo do teste;

🢞 OrderMinStreamerLatency = Envio de Ordem ⭢ Menor latência;

🢞 OrderMaxStreamerLatency = Envio de Ordem ⭢ Maior latência;

🢞 OrderAvgStreamerLatency = Envio de Ordem ⭢ Latência Média.

🢞 OrderMinExchangeIntegrationLatency = Envio Bolsa (roundtrip) ⭢ Menor latência;

🢞 OrderMaxExchangeIntegrationLatency = Envio Bolsa (roundtrip) ⭢ Maior latência;

🢞 OrderAvgExchangeIntegrationLatency = Envio Bolsa (roundtrip) ⭢ Latência Média;

🢞 QuotesMinStreamerLatency = Cotação ⭢ Menor latência;

🢞 QuotesMaxStreamerLatency = Cotação ⭢ Maior latência;

🢞 QuotesAvgStreamerLatency = Cotação ⭢ Latência Média.

▪ Na classe “*LoadTestCommands*” estão configurados as Ordens e Cotações enviadas (necessário ajustar os parâmetros para os testes desejados).

**Atenção**: Uso indevido deste recurso pode causar indisponibilidade do Streamer e OMS devido a carga de requisições excessivas. Na instalação padrão esta funcionalidade está desabilitada (para habilitar esta função configurar o arquivo “*CMA.StreamerFeedAPI.WinForm.Tester.exe.config*” parâmetro “*LoadTest*” com o valor “*S*” → <add key="LoadTest" value="S"/>).

# 4.2) Função Provider ⭢ DBFeed

▪ Funcionalidade exemplo para gravar cotações no banco de dados no padrão CMA BDSQL Quotes;

▪ Exemplos de Connection String:

**🢞 MS/SQL:**

**User ID=abc;Password=123;Server=10.30.0.1;Initial Catalog=bdsql;Persist Security Info=False;**

Observação: Onde “*bdsql*” é o Data Base para gravação das informações.

**🢞 MySQL:**

**Server=10.30.0.1;Port=3306;Database=bdsql;Uid=abc;Pwd=123;SslMode=none;**

Observação: Onde “*bdsql*” é o Data Base para gravação das informações.

**🢞 Oracle:**

**User Id=abc;Password=123;Data Source=10.30.0.1:1521/def;Validate Connection=true;**

Observação: Onde “*def*” é a instância do servidor Oracle.

▪ No repositório do WinForm.Tester estão disponíveis scripts e arquivos do CMA BDSQL Quotes e vários exemplos de arquivos Fluid para os bancos de dados MySQL, MS/SQL e Oracle;

▪ Os templates utilizam o Fluid .Net (engine de template baseado na linguagem de template Liquid), mais detalhes da linguagem de template Liquid em <https://shopify.github.io/liquid/>.

# 4.2.1) Template via Quotes Template

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

▪ Onde:

🢞 Connection String = String para conexão ao banco de dados;

🢞 SourceId | SymbolId = Lista de Id da Origem do Símbolo e Símbolo para gravar na base de dados;

🢞 Provider = Tipo de banco de dados;

🢞 Quotes Template (Fluid) = Template em Fuid para ser executado em cada cotação recebida (vide exemplos abaixo).

▪ Exemplo Quotes Template em **MS/SQL**:

{% for q in Model %}

UPDATE BCSYS\_ATIVO SET ATV\_DAT\_DATAHORA=GETDATE()

{% if q.Data.LastTradePrice != empty %}

,ATV\_DES\_ULT='{{q.Data.LastTradePrice}}'

{% endif %}

{% if q.Data.PrevClosePrice != empty %}

,ATV\_DES\_FEC='{{q.Data.PrevClosePrice}}'

{% endif %}

WHERE ATV\_CODIGO='{{q.Data.Symbol.SymbolID}}'

IF @@ROWCOUNT=0

INSERT INTO BCSYS\_ATIVO (ATV\_CODIGO,ATV\_DAT\_DATAHORA,ATV\_DES\_ULT,ATV\_DES\_FEC)

VALUES ('{{q.Data.Symbol.SymbolID}}',GETDATE(),'{{q.Data.LastTradePrice}}'

,'{{q.Data.PrevClosePrice}}')

{% endfor %}

▪ Exemplo Quotes Template em **MySQL**:

{% for q in Model %}

REPLACE INTO BCSYS\_ATIVO (ATV\_CODIGO,ATV\_DAT\_DATAHORA

{% if q.Data.LastTradePrice != empty %}

,ATV\_DES\_ULT

{% endif %}

{% if q.Data.PrevClosePrice != empty %}

,ATV\_DES\_FEC

{% endif %}

) VALUES ('{{q.Data.Symbol.SymbolID}}',NOW()

{% if q.Data.LastTradePrice != empty %}

,'{{q.Data.LastTradePrice}}'

{% endif %}

{% if q.Data.PrevClosePrice != empty %}

,'{{q.Data.PrevClosePrice}}'

{% endif %}

);

{% endfor %}

Atenção: O comando “*REPLACE*” do MySQL executa primeiro a deleção e em seguida a inclusão, ou seja, este comportamento pode não ser desejado em casos em que as colunas não são atualizadas, neste cenário podemos utilizar uma “*procedure*” para realizar a atualização de cotação, segue abaixo exemplo:

Procedure:

DELIMITER $$

CREATE PROCEDURE SetBCSYS\_ATIVO(

IN pATV\_CODIGO VARCHAR(20)

,pATV\_DES\_ULT VARCHAR(50)

,pATV\_DES\_FEC VARCHAR(50))

BEGIN

UPDATE BCSYS\_ATIVO

SET ATV\_DAT\_DATAHORA=NOW()

,ATV\_DES\_ULT=IFNULL(pATV\_DES\_ULT,ATV\_DES\_ULT)

,ATV\_DES\_FEC=IFNULL(pATV\_DES\_FEC,ATV\_DES\_FEC)

WHERE ATV\_CODIGO=pATV\_CODIGO;

IF ROW\_COUNT()=0 THEN

INSERT INTO BCSYS\_ATIVO (ATV\_CODIGO,ATV\_DAT\_DATAHORA,ATV\_DES\_ULT,ATV\_DES\_FEC)

VALUES (pATV\_CODIGO,NOW(),pATV\_DES\_ULT,pATV\_DES\_FEC);

END IF;

END$$

DELIMITER ;

Template:

{% for q in Model %}

CALL SetBCSYS\_ATIVO ('{{q.Data.Symbol.SymbolID}}'

{% if q.Data.LastTradePrice != empty %}

,'{{q.Data.LastTradePrice}}'

{% else %}

,NULL

{% endif %}

{% if q.Data.PrevClosePrice != empty %}

,'{{q.Data.PrevClosePrice}}'

{% else %}

,NULL

{% endif %}

);

{% endfor %}

▪ Exemplo Quotes Template em **Oracle**:

BEGIN

{% for q in Model %}

UPDATE BCSYS\_ATIVO SET ATV\_DAT\_DATAHORA=SYSDATE

{% if q.Data.LastTradePrice != empty %}

,ATV\_DES\_ULT='{{q.Data.LastTradePrice}}'

{% endif %}

{% if q.Data.PrevClosePrice != empty %}

,ATV\_DES\_FEC='{{q.Data.PrevClosePrice}}'

{% endif %}

WHERE ATV\_CODIGO='{{q.Data.Symbol.SymbolID}}';

IF SQL%NOTFOUND THEN

INSERT INTO BCSYS\_ATIVO (ATV\_CODIGO,ATV\_DAT\_DATAHORA,ATV\_DES\_ULT,ATV\_DES\_FEC)

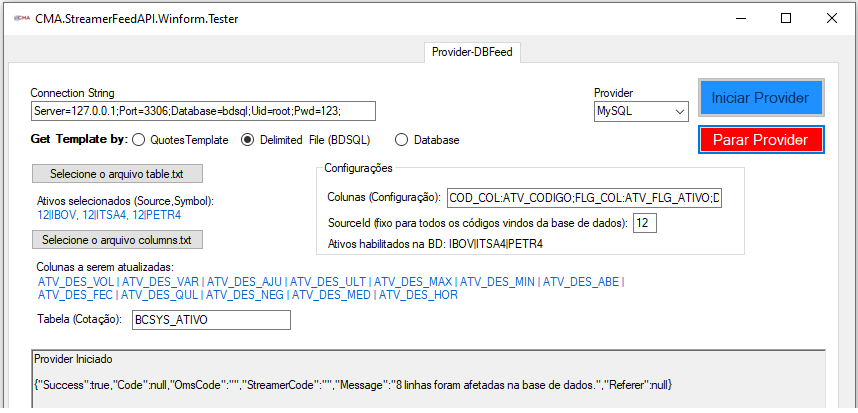
VALUES ('{{q.Data.Symbol.SymbolID}}',SYSDATE,'{{q.Data.LastTradePrice}}','{{q.Data.PrevClosePrice}}');

END IF;

{% endfor %}

END;

# 4.2.2) Template via Delimited File (BDSQL)



▪ Onde:

🢞 Connection String = String para conexão ao banco de dados;

🢞 Provider = Tipo de banco de dados;

🢞 Arquivo table.txt = Arquivo table.txt (padrão do CMA BDSQL Quotes) com a lista de códigos;

🢞 Arquivo columns.txt = Arquivo columns.txt (padrão do CMA BDSQL Quotes) com a lista de colunas utilizadas;

🢞 Colunas (Configuração) = Colunas de Código (COD\_COL), Flag Ativo (FLG\_COL) e Data/Hora (DTM\_COL), segue padrão do CMA BDSQL Quotes:

- COD\_COL:ATV\_CODIGO;

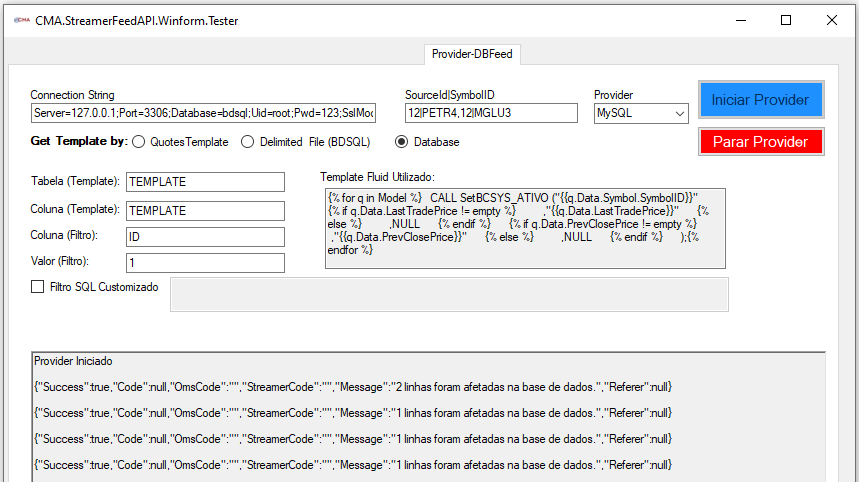
- FLG\_COL:ATV\_FLG\_ATIVO;

- DTM\_COL:ATV\_DAT\_DATAHORA.

🢞 SourceId (fixo para os códigos vindos da base) = Id da Origem do Símbolo para os códigos habilitados recuperados da base de dados;

🢞 Tabela (Cotação): Nome da tabela que será atualizada com cotações (BCSYS\_ATIVO padrão CMA BDSQL Quotes).

# 4.2.3) Template via Database



▪ Onde:

🢞 Connection String = String para conexão ao banco de dados;

🢞 SourceId | SymbolId = Lista de Id da Origem do Símbolo e Símbolo (separados por “|”) para gravar na base de dados delimitados por “,” (exemplo: 12|PETR4,12|MGLU3);

🢞 Provider = Tipo de banco de dados;

🢞 Tabela (Template) = Nome da tabela onde está o template desejado;

🢞 Coluna (Template) = Coluna da tabela onde está o texto do template desejado;

🢞 Coluna (Filtro) = Coluna onde vai estar o filtro;

🢞 Valor (Filtro) = Valor do filtro para chegar no registro do template (se retornar mais de um apenas o primeiro será utilizado);

🢞 Filtro SQL Customizado = Query para selecionar o template desejado (exemplo: select TEMPLATE from TEMPLATE where ID=1).

Observação: Para exemplos de templates (Fluid) vide item “*4.2.1*”.