Recto Verso Core Smart Contract Functions

1. InitializeMarket

Deskripsi:

Instruksi **InitializeMarket** memungkinkan pengguna untuk membuat pasar baru di platform **Recto Verso**. Instruksi ini dirancang agar dapat diakses oleh semua pengguna, terutama para developer yang ingin memperkenalkan dan mengelola pasar token mereka sendiri dengan mudah. Dengan instruksi ini, pengguna dapat mendefinisikan parameter-parameter pasar seperti pasangan token, ukuran tick, jumlah lot dasar, dan taker fee. Selain itu, untuk memulai pasar, pengguna harus membayar fee dalam bentuk token **RESO** dan menyediakan likuiditas awal sesuai dengan ketentuan minimal yang telah ditetapkan oleh sistem.

Parameter Utama:

- **marketSizeParams**:
  - Parameter yang menentukan ukuran pasar, termasuk:
    - **bidsSize**: Ukuran untuk sisi bids (pembelian) dalam order book.
    - **asksSize**: Ukuran untuk sisi asks (penjualan) dalam order book.
    - **numSeats**: Jumlah seat yang tersedia di pasar, yang menentukan berapa banyak peserta yang dapat berpartisipasi.
- **numQuoteLotsPerQuoteUnit**:
  - Jumlah lot kutipan per unit kutipan. Parameter ini menentukan bagaimana kutipan token diukur dan diperdagangkan di pasar.
- **tickSizeInQuoteLotsPerBaseUnit**:
  - Ukuran tick dalam lot kutipan per unit dasar. Ini mengatur increment harga terkecil yang dapat diterima oleh order di pasar.
- **numBaseLotsPerBaseUnit**:
  - Jumlah lot dasar per unit dasar. Ini menentukan bagaimana lot dasar diukur dan diperdagangkan di pasar.
- **takerFeeBps**:
  - Fee yang dibebankan kepada taker (pihak yang mengambil likuiditas dari order book) dalam basis poin. Ini adalah bagian dari biaya transaksi yang akan dibayar oleh taker saat mengeksekusi order di pasar.
- **rawBaseUnitsPerBaseUnit** (opsional):
  - Unit dasar mentah per unit dasar, jika diperlukan untuk menyesuaikan unit perdagangan di pasar.
- **initialLiquidityBase**:
  - Jumlah likuiditas awal untuk token dasar yang harus disediakan oleh pengguna saat membuat pasar. Likuiditas ini harus memenuhi jumlah minimum yang telah ditetapkan oleh sistem.
- **initialLiquidityQuote**:
  - Jumlah likuiditas awal untuk token kutipan yang juga harus disediakan oleh pengguna. Ini adalah syarat untuk memastikan bahwa pasar memiliki likuiditas yang cukup untuk mendukung aktivitas perdagangan.

Fungsi Utama:

Instruksi **InitializeMarket** berfungsi untuk memberikan pengguna kemampuan untuk membuat pasar baru dengan parameter yang dapat disesuaikan, sambil memastikan bahwa pasar memiliki likuiditas yang memadai dan biaya pembuatan yang adil melalui pembayaran fee dalam token **RESO**. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Akses Terbuka untuk Semua Pengguna**:
   - **InitializeMarket** dapat diakses oleh semua pengguna, memungkinkan siapa saja, terutama developer, untuk membuat dan mengelola pasar token mereka sendiri. Ini mendorong partisipasi yang lebih luas di platform dan memungkinkan pertumbuhan ekosistem yang lebih dinamis.
2. **Pembayaran Fee dalam Token RESO**:
   - Untuk membuat pasar, pengguna harus membayar fee dalam bentuk token **RESO**. Ini memastikan bahwa setiap pasar yang dibuat memberikan kontribusi kepada ekosistem melalui pembayaran fee, yang dapat digunakan untuk mendukung pengembangan platform lebih lanjut.
3. **Penyediaan Likuiditas Awal**:
   - Pengguna yang membuat pasar harus menyediakan likuiditas awal untuk kedua token yang diperdagangkan (base dan quote). Likuiditas ini penting untuk memastikan bahwa pasar memiliki cukup dana untuk mendukung perdagangan dan menjaga stabilitas harga. Sistem juga menetapkan jumlah minimum likuiditas yang harus dipenuhi, sehingga pasar baru tidak kekurangan likuiditas.
4. **Kontrol terhadap Parameter Pasar**:
   - Pengguna memiliki kontrol penuh terhadap parameter-parameter pasar seperti ukuran tick, jumlah lot, dan fee taker. Ini memungkinkan fleksibilitas dalam mendesain pasar yang sesuai dengan kebutuhan dan strategi perdagangan mereka.
5. **Peningkatan Likuiditas dan Aktivitas Pasar**:
   - Dengan persyaratan likuiditas awal, **InitializeMarket** membantu memastikan bahwa setiap pasar baru memiliki cukup dana untuk mendukung aktivitas perdagangan sejak awal. Ini membantu mencegah masalah likuiditas rendah yang bisa menghambat fungsi pasar.

Contoh Penggunaan:

- Seorang developer yang telah mengembangkan token baru ingin meluncurkan pasar untuk token tersebut di platform **Recto Verso**. Dengan menggunakan **InitializeMarket**, mereka dapat membuat pasar dengan parameter yang disesuaikan, seperti ukuran tick dan jumlah lot, serta menyertakan likuiditas awal dalam token dasar dan kutipan. Setelah membayar fee dalam token **RESO**, pasar baru siap digunakan oleh trader lain di platform.
- Dalam skenario lain, sebuah komunitas yang mendukung token tertentu ingin membuat pasar baru di mana token tersebut bisa diperdagangkan. Mereka mengumpulkan dana untuk menyediakan likuiditas awal dan menggunakan **InitializeMarket** untuk meluncurkan pasar baru, memberikan anggota komunitas mereka akses ke perdagangan token yang mereka dukung.

2. ModifyMarketState

Deskripsi:

Instruksi **ModifyMarketState** memungkinkan otoritas pasar atau entitas yang berwenang untuk mengubah status atau kondisi dari pasar tertentu di platform **Recto Verso**. Instruksi ini penting untuk menjaga fleksibilitas dan kontrol atas operasi pasar, memungkinkan pengelola pasar untuk merespons berbagai kondisi pasar atau kebutuhan strategis dengan cepat. Status pasar yang dapat diubah mencakup berbagai aspek seperti apakah pasar aktif, hanya menerima order post-only, dalam keadaan jeda, atau ditutup sepenuhnya. Dengan instruksi ini, pengelola pasar dapat mengoptimalkan operasi pasar dan melindungi pengguna serta likuiditas yang ada di dalamnya.

Parameter Utama:

- **market**:
  - Pasar yang statusnya akan diubah. Ini mengidentifikasi pasar spesifik yang statusnya akan dimodifikasi melalui instruksi ini.
- **marketStatus**:
  - Status baru yang akan diterapkan ke pasar. Status yang mungkin termasuk:
    - **Active**: Pasar berada dalam kondisi aktif, menerima semua jenis order dan eksekusi berjalan seperti biasa.
    - **PostOnly**: Pasar hanya menerima order post-only, yang berarti order tidak akan dieksekusi langsung dan hanya akan ditambahkan ke order book.
    - **Paused**: Pasar dijeda sementara, menghentikan semua aktivitas perdagangan tanpa membatalkan order yang ada di order book.
    - **Closed**: Pasar ditutup, menghentikan semua aktivitas perdagangan dan biasanya melibatkan pembatalan semua order yang tersisa.
    - **Tombstoned**: Pasar dihapus secara permanen dan tidak dapat diakses lagi.

Fungsi Utama:

Instruksi **ModifyMarketState** berfungsi untuk memberikan fleksibilitas kepada otoritas pasar dalam mengelola kondisi pasar berdasarkan kebutuhan operasional dan situasi pasar yang ada. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pengelolaan Dinamis atas Pasar**:
   - **ModifyMarketState** memungkinkan otoritas pasar untuk mengubah status pasar secara dinamis sesuai dengan kebutuhan operasional atau respons terhadap kondisi pasar tertentu. Misalnya, jika terjadi volatilitas pasar yang ekstrem, pasar dapat dijeda untuk melindungi pengguna.
2. **Perlindungan Likuiditas dan Pengguna**:
   - Dengan kemampuan untuk menjeda atau menutup pasar, instruksi ini memungkinkan pengelola pasar untuk mengambil tindakan proaktif dalam melindungi likuiditas dan melindungi pengguna dari kondisi perdagangan yang tidak diinginkan.
3. **Kontrol Operasional**:
   - Instruksi ini memberikan kontrol penuh kepada otoritas pasar untuk mengelola bagaimana dan kapan pasar beroperasi. Misalnya, mereka dapat menetapkan pasar ke status post-only selama periode pembaruan teknis atau untuk mengelola kondisi likuiditas.
4. **Penutupan dan Penghapusan Pasar**:
   - Dalam situasi di mana pasar tidak lagi diperlukan atau harus dihentikan, **ModifyMarketState** memungkinkan otoritas untuk menutup atau menghapus pasar dengan aman dan efisien,

memastikan bahwa semua order dibatalkan dan tidak ada aktivitas perdagangan lebih lanjut yang dapat terjadi.

5. **Penyelarasan dengan Kebijakan atau Strategi**:
   - Instruksi ini juga memungkinkan pengelola pasar untuk menyesuaikan status pasar agar selaras dengan kebijakan internal atau strategi yang diterapkan oleh entitas yang mengelola pasar, seperti membatasi perdagangan dalam kondisi tertentu atau memfasilitasi pengenalan fitur baru.

Contoh Penggunaan:

- Dalam situasi di mana pasar mengalami volatilitas ekstrem yang dapat menyebabkan risiko bagi trader, pengelola pasar dapat menggunakan **ModifyMarketState** untuk menjeda pasar, menghentikan sementara semua perdagangan hingga kondisi stabil.
- Jika pasar perlu dihentikan untuk perawatan atau pembaruan teknis, pengelola dapat menetapkan status pasar ke **PostOnly**, sehingga order baru masih bisa ditambahkan ke order book tetapi tidak ada eksekusi yang akan terjadi hingga pasar kembali ke status **Active**.
- Sebuah pasar yang sudah tidak lagi aktif atau digunakan dapat dihapus secara permanen dengan mengubah statusnya menjadi **Tombstoned** menggunakan **ModifyMarketState**, memastikan bahwa pasar tersebut tidak lagi muncul dalam daftar pasar yang tersedia dan semua aktivitas perdagangan dihentikan.

3. InstantOrder

Deskripsi:

Instruksi **InstantOrder** adalah salah satu fungsi utama di platform **Recto Verso** yang memungkinkan pengguna untuk mengeksekusi order secara instan berdasarkan kondisi pasar saat ini. Tidak seperti order limit, yang menunggu hingga harga yang diinginkan tercapai, **InstantOrder** segera dieksekusi dengan harga terbaik yang tersedia di order book pada saat perintah dijalankan. Instruksi ini sangat berguna dalam situasi di mana kecepatan eksekusi lebih penting daripada harga tertentu.

Parameter Utama:

- **side**:
  - Menentukan apakah pengguna ingin membeli (**Bid**) atau menjual (**Ask**) aset. Parameter ini mengarahkan sistem untuk mencari pasangan order yang sesuai di sisi berlawanan dari order book.
- **price_in_ticks**:
  - Harga yang ditentukan dalam bentuk ticks. Meskipun ini merupakan batas harga yang diinginkan pengguna, dalam **InstantOrder**, order ini mungkin dieksekusi pada harga terbaik yang tersedia jika tidak ada order yang cocok pada harga yang tepat.
- **num_base_lots**:
  - Jumlah lot dasar yang ingin diperdagangkan oleh pengguna. Ini menentukan volume aset yang akan dibeli atau dijual.
- **client_order_id**:
  - ID unik yang ditetapkan oleh pengguna atau sistem untuk melacak dan mengidentifikasi order tertentu. Ini membantu dalam manajemen order dan pelaporan.

- **self_trade_behavior**:
  - Menentukan bagaimana sistem harus menangani situasi di mana order pengguna mungkin dipasangkan dengan order lain yang mereka miliki (self-trade). Opsi yang tersedia termasuk **Abort** (membatalkan order jika self-trade terdeteksi), **CancelProvide** (membatalkan order pasif yang sudah ada), dan **DecrementTake** (mengurangi ukuran order baru untuk menghindari self-trade).
- **match_limit** (opsional):
  - Batas jumlah match yang diizinkan dalam satu eksekusi order. Ini berguna untuk mengontrol seberapa banyak eksekusi yang terjadi dalam satu transaksi.
- **last_valid_slot** (opsional):
  - Slot terakhir yang valid di mana order masih bisa dieksekusi. Setelah slot ini, order tidak akan dijalankan lagi, memberikan batas waktu untuk validitas order.
- **last_valid_unix_timestamp_in_seconds** (opsional):
  - Waktu Unix dalam detik terakhir di mana order masih valid untuk dieksekusi. Setelah waktu ini berlalu, order tidak akan diproses.
- **fail_silently_on_insufficient_funds**:
  - Menentukan apakah sistem akan gagal secara diam-diam (tanpa pemberitahuan atau rollback) jika dana yang tersedia tidak mencukupi untuk mengeksekusi order. Ini memberikan fleksibilitas untuk mengelola situasi di mana dana pengguna mungkin tidak cukup.

Fungsi Utama:

Instruksi **InstantOrder** berfungsi untuk mengeksekusi order secara cepat dan efisien berdasarkan kondisi pasar saat itu. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Eksekusi Cepat di Pasar**:
   - **InstantOrder** memungkinkan pengguna untuk segera membeli atau menjual aset tanpa harus menunggu harga tertentu tercapai. Ini sangat berguna dalam situasi di mana kecepatan lebih penting daripada harga tertentu, seperti dalam perdagangan arbitrase atau ketika merespons perubahan pasar yang cepat.
2. **Kontrol Terhadap Self-Trade**:
   - Parameter **self_trade_behavior** memberikan kontrol penuh kepada pengguna untuk mengelola bagaimana self-trade ditangani, memastikan bahwa strategi perdagangan pengguna tetap terlindungi dan tidak terganggu oleh eksekusi yang tidak diinginkan.
3. **Batas Waktu Validitas Order**:
   - Dengan parameter **last_valid_slot** dan **last_valid_unix_timestamp_in_seconds**, pengguna dapat mengatur batas waktu validitas order mereka, memastikan bahwa order tidak akan dijalankan di luar waktu yang diharapkan atau diinginkan.
4. **Penanganan Kegagalan yang Fleksibel**:
   - Parameter **fail_silently_on_insufficient_funds** memberikan fleksibilitas tambahan dalam menangani situasi di mana dana pengguna tidak mencukupi. Alih-alih menggagalkan seluruh transaksi atau menyebabkan error yang mengganggu, sistem dapat diam-diam mengabaikan order yang gagal tanpa mempengaruhi pengalaman pengguna.

Contoh Penggunaan:

- Seorang trader melihat peluang pasar yang sempit dan membutuhkan eksekusi cepat untuk memanfaatkan perbedaan harga antara dua aset. Dengan menggunakan **InstantOrder**, trader dapat

segera mengeksekusi pembelian atau penjualan pada harga terbaik yang tersedia di pasar.
- Dalam situasi di mana pasar sangat volatil dan harga berubah dengan cepat, **InstantOrder** memungkinkan pengguna untuk mengeksekusi order mereka dengan cepat sebelum kondisi pasar berubah lagi.

4. PlaceLimitOrder

Deskripsi:

Instruksi **PlaceLimitOrder** memungkinkan pengguna untuk menempatkan order limit di pasar. Tidak seperti **InstantOrder**, di mana eksekusi terjadi segera pada harga pasar terbaik, **PlaceLimitOrder** hanya akan dieksekusi ketika harga pasar mencapai atau melewati harga yang ditentukan oleh pengguna. Ini memberikan kontrol lebih besar kepada pengguna atas harga di mana mereka bersedia membeli atau menjual aset, dan sangat berguna dalam strategi perdagangan di mana harga tertentu dianggap penting.

Parameter Utama:

- **side**:
  - Menentukan apakah pengguna ingin membeli (**Bid**) atau menjual (**Ask**) aset. Parameter ini mengarahkan sistem untuk menempatkan order pada sisi yang sesuai di order book.
- **price_in_ticks**:
  - Harga aset yang diinginkan dalam tick. Tick adalah unit terkecil dari harga yang ditentukan dalam sistem. Order hanya akan dieksekusi jika harga pasar mencapai atau melewati harga ini.
- **num_base_lots**:
  - Jumlah lot dasar yang ingin diperdagangkan oleh pengguna. Ini menentukan volume aset yang akan dibeli atau dijual ketika kondisi harga terpenuhi.
- **self_trade_behavior**:
  - Menentukan bagaimana sistem harus menangani situasi di mana order pengguna mungkin dipasangkan dengan order lain yang mereka miliki (self-trade). Opsi yang tersedia termasuk:
    - **Abort**: Membatalkan order jika self-trade terdeteksi.
    - **CancelProvide**: Membatalkan order pasif yang sudah ada jika self-trade terdeteksi.
    - **DecrementTake**: Mengurangi ukuran order baru untuk menghindari self-trade.
- **match_limit** (opsional):
  - Batas jumlah match yang diizinkan dalam satu eksekusi order. Ini berguna untuk mengontrol seberapa banyak eksekusi yang terjadi dalam satu transaksi.
- **client_order_id**:
  - ID unik yang ditetapkan oleh pengguna atau sistem untuk melacak dan mengidentifikasi order tertentu. Ini membantu dalam manajemen order dan pelaporan.
- **last_valid_slot** (opsional):
  - Slot terakhir yang valid di mana order masih bisa dieksekusi. Setelah slot ini, order tidak akan dijalankan lagi, memberikan batas waktu untuk validitas order.
- **last_valid_unix_timestamp_in_seconds** (opsional):
  - Waktu Unix dalam detik terakhir di mana order masih valid untuk dieksekusi. Setelah waktu ini berlalu, order tidak akan diproses.
- **fail_silently_on_insufficient_funds**:

- Menentukan apakah sistem akan gagal secara diam-diam (tanpa pemberitahuan atau rollback) jika dana yang tersedia tidak mencukupi untuk mengeksekusi order. Ini memberikan fleksibilitas untuk mengelola situasi di mana dana pengguna mungkin tidak cukup.

Fungsi Utama:

Instruksi **PlaceLimitOrder** berfungsi untuk menempatkan order di order book yang hanya akan dieksekusi jika kondisi harga tertentu terpenuhi. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Kontrol Penuh atas Harga Eksekusi**:
   - **PlaceLimitOrder** memberikan pengguna kontrol penuh atas harga di mana mereka ingin mengeksekusi order mereka. Order tidak akan dieksekusi kecuali harga pasar mencapai atau melebihi harga yang ditentukan oleh pengguna, memungkinkan strategi perdagangan yang lebih presisi.
2. **Likuiditas Pasar**:
   - Dengan menempatkan order limit, pengguna berkontribusi pada likuiditas pasar. Order yang ditempatkan di order book menambah kedalaman pasar, yang dapat membantu stabilitas harga dan memberikan lebih banyak pilihan bagi trader lain.
3. **Penghindaran Eksekusi yang Tidak Diinginkan**:
   - Pengguna dapat menggunakan **PlaceLimitOrder** untuk memastikan bahwa mereka tidak membeli atau menjual aset pada harga yang tidak menguntungkan. Ini sangat penting dalam kondisi pasar yang volatil di mana harga dapat berubah dengan cepat.
4. **Kontrol Terhadap Self-Trade**:
   - Parameter **self_trade_behavior** memberikan kontrol kepada pengguna untuk mengelola bagaimana self-trade ditangani, memastikan bahwa order mereka tidak secara tidak sengaja dipasangkan dengan order lain yang mereka miliki.
5. **Batas Waktu Validitas Order**:
   - Dengan parameter **last_valid_slot** dan **last_valid_unix_timestamp_in_seconds**, pengguna dapat mengatur batas waktu validitas order mereka, memastikan bahwa order tidak akan dijalankan di luar waktu yang diharapkan atau diinginkan.
6. **Penanganan Kegagalan yang Fleksibel**:
   - Parameter **fail_silently_on_insufficient_funds** memberikan fleksibilitas tambahan dalam menangani situasi di mana dana pengguna tidak mencukupi. Alih-alih menggagalkan seluruh transaksi atau menyebabkan error yang mengganggu, sistem dapat diam-diam mengabaikan order yang gagal tanpa mempengaruhi pengalaman pengguna.

Contoh Penggunaan:

- Seorang trader percaya bahwa harga aset akan turun dalam beberapa jam ke depan dan ingin membeli aset tersebut hanya jika harga mencapai titik tertentu. Dengan menggunakan **PlaceLimitOrder**, mereka dapat menempatkan order beli pada harga target dan tidak perlu terus-menerus memantau pasar. Jika harga mencapai titik yang diinginkan, order akan dieksekusi; jika tidak, order akan tetap berada di order book.
- Dalam situasi di mana pasar sangat volatil, seorang trader mungkin ingin menjual aset mereka jika harga mencapai level tertentu yang lebih tinggi. Dengan **PlaceLimitOrder**, mereka dapat menempatkan order jual dan yakin bahwa aset mereka hanya akan dijual pada atau di atas harga yang mereka anggap cukup tinggi.

5. PlaceMultiplePendingOrders

Deskripsi:

Instruksi **PlaceMultiplePendingOrders** memungkinkan pengguna untuk menempatkan beberapa order pending di order book secara bersamaan. Order pending adalah order yang hanya akan dieksekusi jika kondisi tertentu terpenuhi, seperti harga mencapai level yang ditentukan. Instruksi ini sangat berguna bagi pengguna yang ingin menerapkan strategi perdagangan yang melibatkan penempatan beberapa order pada berbagai level harga tanpa harus memasukkan order satu per satu. Dengan instruksi ini, pengguna dapat mengelola portofolio mereka dengan lebih efisien, memanfaatkan peluang pasar yang berbeda-beda secara bersamaan.

Parameter Utama:

- **bids**:
  - Daftar order bid (pembelian) yang ingin ditempatkan. Setiap item dalam daftar ini mencakup:
    - **price_in_ticks**: Harga yang diinginkan dalam bentuk ticks untuk order bid.
    - **size_in_base_lots**: Ukuran lot dasar yang ingin dibeli pada harga tersebut.
    - **last_valid_slot** (opsional): Slot terakhir yang valid untuk eksekusi order. Setelah slot ini, order tidak akan dieksekusi lagi.
    - **last_valid_unix_timestamp_in_seconds** (opsional): Waktu Unix dalam detik terakhir yang valid untuk eksekusi order. Setelah waktu ini, order tidak akan diproses.
- **asks**:
  - Daftar order ask (penjualan) yang ingin ditempatkan. Struktur parameter mirip dengan **bids**, dengan harga dan ukuran yang ditentukan oleh pengguna untuk penjualan aset.
- **client_order_id** (opsional):
  - ID unik yang ditetapkan oleh pengguna atau sistem untuk melacak dan mengidentifikasi setiap order dalam daftar. Ini membantu dalam manajemen order dan pelaporan.
- **failedMultipleLimitOrderBehavior**:
  - Menentukan perilaku sistem jika salah satu order dalam daftar gagal dieksekusi. Opsi yang tersedia termasuk:
    - **FailOnInsufficientFundsAndAmendOnCross**: Jika dana tidak mencukupi atau order tereksekusi dengan kondisi self-trade, order yang gagal akan diabaikan dan sistem akan melanjutkan dengan order lainnya.

Fungsi Utama:

Instruksi **PlaceMultiplePendingOrders** berfungsi untuk memungkinkan pengguna menempatkan beberapa order limit secara efisien, dengan fleksibilitas untuk mengatur berbagai parameter yang relevan. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Penempatan Order yang Efisien**:
   - Dengan **PlaceMultiplePendingOrders**, pengguna dapat menempatkan beberapa order pada berbagai level harga dalam satu langkah. Ini sangat efisien dibandingkan harus memasukkan setiap order secara individual, terutama dalam strategi yang melibatkan banyak order kecil.

2. **Strategi Perdagangan Kompleks**:
   - Instruksi ini memungkinkan pengguna untuk menerapkan strategi perdagangan yang lebih kompleks, seperti grid trading atau laddering, di mana beberapa order ditempatkan pada interval harga yang berbeda-beda untuk memanfaatkan fluktuasi pasar.
3. **Manajemen Risiko**:
   - Pengguna dapat menggunakan parameter seperti **last_valid_slot** dan **last_valid_unix_timestamp_in_seconds** untuk mengontrol risiko dengan memastikan bahwa order hanya berlaku untuk periode waktu tertentu. Ini membantu mencegah eksekusi order yang tidak diinginkan di masa depan ketika kondisi pasar mungkin sudah berubah.
4. **Kontrol Terhadap Kegagalan Order**:
   - Dengan parameter **failedMultipleLimitOrderBehavior**, pengguna dapat menentukan bagaimana sistem harus menangani situasi di mana satu atau lebih order gagal. Ini memberikan fleksibilitas dalam manajemen order dan membantu mengurangi dampak negatif dari kegagalan eksekusi.
5. **Likuiditas Pasar**:
   - Dengan menempatkan beberapa order pada berbagai harga, pengguna berkontribusi pada likuiditas pasar, membuat lebih banyak peluang perdagangan bagi trader lain dan membantu menjaga stabilitas harga.

Contoh Penggunaan:

- Seorang trader yang menerapkan strategi grid trading mungkin ingin menempatkan beberapa order beli dan jual pada berbagai level harga yang telah ditentukan sebelumnya. Dengan menggunakan **PlaceMultiplePendingOrders**, mereka dapat menempatkan semua order tersebut dalam satu langkah, menghemat waktu dan usaha.
- Dalam skenario lain, seorang trader mungkin ingin menempatkan beberapa order limit untuk memanfaatkan fluktuasi harga dalam kisaran tertentu. Mereka dapat mengatur parameter **last_valid_slot** atau **last_valid_unix_timestamp_in_seconds** untuk memastikan bahwa order tersebut hanya berlaku selama waktu tertentu, mencegah eksekusi order di luar waktu yang diinginkan.

6. CancelOrder

Deskripsi:

Instruksi **CancelOrder** memungkinkan pengguna untuk membatalkan satu atau lebih order yang telah mereka tempatkan di order book berdasarkan kriteria tertentu. Instruksi ini sangat penting dalam situasi di mana pengguna ingin mengubah strategi perdagangan mereka, menghentikan order yang tidak lagi sesuai dengan kondisi pasar, atau menghindari risiko yang tidak diinginkan.

Parameter Utama:

- **side**:
  - Menentukan apakah order yang dibatalkan adalah **Bid** (order beli) atau **Ask** (order jual). Ini membantu sistem untuk mengidentifikasi sisi mana dari order book yang harus dicari untuk menemukan order yang akan dibatalkan.

- **tickLimit**:
  - Merupakan batas harga dalam tick yang menentukan hingga harga mana order akan dibatalkan. TickLimit memungkinkan pengguna untuk membatalkan order yang hanya berada di rentang harga tertentu, sehingga memberikan fleksibilitas dalam pengelolaan order.
- **numOrdersToSearch**:
  - Parameter ini menentukan jumlah maksimum order yang akan dicari dalam order book. Misalnya, pengguna mungkin hanya ingin mencari dan membatalkan sejumlah order tertentu, bukan seluruh order book. Parameter ini membantu dalam mengontrol efisiensi pencarian, terutama di pasar dengan likuiditas tinggi.
- **numOrdersToCancel**:
  - Menentukan jumlah maksimum order yang akan dibatalkan. Pengguna mungkin tidak ingin membatalkan semua order yang ditemukan, melainkan hanya sejumlah order tertentu. Parameter ini memungkinkan kontrol yang lebih presisi terhadap proses pembatalan.

Fungsi Utama:

Instruksi **CancelOrder** berfungsi untuk membatalkan order yang telah ditempatkan di order book. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Menghindari Eksekusi Order yang Tidak Diinginkan**:
   - Pasar bisa sangat dinamis, dan kondisi harga dapat berubah dengan cepat. **CancelOrder** memungkinkan pengguna untuk dengan cepat membatalkan order yang tidak lagi sesuai dengan strategi atau harapan mereka sebelum order tersebut dieksekusi.
2. **Pengelolaan Risiko**:
   - Dengan membatalkan order yang tidak lagi relevan, pengguna dapat mengurangi potensi risiko yang dihadapi. Misalnya, jika pengguna menempatkan order beli dengan harga yang lebih tinggi dari harga pasar saat ini, mereka mungkin ingin membatalkan order tersebut untuk menghindari overpaying.
3. **Pembebasan Dana yang Terkunci**:
   - Saat order ditempatkan di order book, sejumlah dana akan dikunci sebagai jaminan. Dengan membatalkan order melalui **CancelOrder**, dana yang terkunci akan dilepaskan dan tersedia kembali untuk digunakan dalam perdagangan lainnya.
4. **Pemeliharaan Order Book**:
   - Instruksi ini membantu menjaga kebersihan dan relevansi order book dengan memungkinkan pengguna untuk membatalkan order yang sudah tidak relevan atau yang tidak akan pernah terpenuhi. Ini juga membantu menghindari penumpukan order yang tidak aktif di order book.
5. **Fleksibilitas dalam Pembatalan**:
   - Dengan parameter seperti **tickLimit** dan **numOrdersToSearch**, pengguna memiliki fleksibilitas untuk menargetkan pembatalan order secara selektif, berdasarkan kondisi pasar yang spesifik dan strategi mereka.

Contoh Penggunaan:

- Seorang trader menempatkan beberapa order beli di bawah harga pasar saat ini dengan harapan harga akan turun. Namun, ketika harga mulai naik, trader tersebut memutuskan untuk membatalkan order-order ini menggunakan **CancelOrder** dengan parameter **side** sebagai Bid dan **tickLimit** yang sesuai dengan rentang harga order tersebut.

- Dalam situasi pasar yang bergerak cepat, trader mungkin ingin membatalkan order yang belum tereksekusi untuk menghindari perubahan harga yang cepat. Dengan **numOrdersToCancel**, trader dapat membatalkan hanya sejumlah order tertentu, meninggalkan yang lainnya untuk dieksekusi di masa mendatang jika kondisi pasar berubah kembali.

7. CancelAllOrders

Deskripsi:

Instruksi **CancelAllOrders** memungkinkan pengguna untuk membatalkan semua order yang telah mereka tempatkan di order book dalam satu pasar tertentu. Ini adalah alat yang sangat berguna ketika pengguna ingin dengan cepat menghapus semua komitmen mereka di pasar, misalnya, dalam situasi pasar yang tiba-tiba berubah atau ketika mereka ingin menutup semua posisi terbuka. Dengan membatalkan semua order secara sekaligus, pengguna dapat segera mengurangi eksposur mereka terhadap risiko yang tidak diinginkan tanpa harus membatalkan order satu per satu.

Parameter Utama:

Instruksi **CancelAllOrders** biasanya tidak memerlukan parameter tambahan karena fungsinya adalah untuk membatalkan semua order yang terkait dengan akun pengguna dalam pasar tertentu. Namun, beberapa implementasi dapat memiliki opsi atau parameter tambahan untuk mempersempit cakupan instruksi ini (misalnya, hanya membatalkan order di sisi tertentu dari order book). Pada umumnya, berikut adalah parameter yang mungkin terkait:

- **market** (opsional):
  - Pasar di mana semua order pengguna akan dibatalkan. Jika tidak ditentukan, biasanya mencakup semua order di pasar yang diakses oleh pengguna saat instruksi dipanggil.
- **side** (opsional):
  - Menentukan apakah instruksi akan membatalkan semua order di sisi **Bid** (beli) atau **Ask** (jual). Jika tidak ditentukan, instruksi akan membatalkan semua order di kedua sisi.

Fungsi Utama: Instruksi **CancelAllOrders** berfungsi untuk membatalkan semua order pengguna yang ada di order book untuk pasar tertentu. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pengurangan Risiko Secara Cepat**:
   - **CancelAllOrders** memungkinkan pengguna untuk segera mengurangi eksposur mereka terhadap pasar dengan membatalkan semua order terbuka. Ini sangat penting dalam kondisi pasar yang volatil di mana perubahan harga yang cepat dapat menyebabkan kerugian signifikan jika order yang tidak diinginkan tetap terbuka.
2. **Efisiensi dalam Manajemen Order**:
   - Instruksi ini memberikan cara yang cepat dan efisien untuk mengelola order dalam situasi di mana pengguna ingin keluar dari pasar atau menarik semua likuiditas yang mereka tawarkan. Dengan satu instruksi, semua order dapat dibatalkan tanpa perlu mengidentifikasi dan membatalkan setiap order secara manual.
3. **Pengelolaan Likuiditas**:

    - Dengan membatalkan semua order, pengguna dapat dengan cepat menarik likuiditas dari pasar. Ini bisa menjadi bagian dari strategi untuk mengurangi dampak pada harga pasar atau untuk menunggu kondisi yang lebih baik sebelum kembali menempatkan order baru.
4. **Menghentikan Strategi Perdagangan**:
    - **CancelAllOrders** adalah alat yang berguna bagi pengguna yang ingin segera menghentikan strategi perdagangan mereka. Misalnya, jika suatu strategi terbukti tidak menguntungkan atau jika terjadi kejadian yang tidak terduga di pasar, pengguna dapat membatalkan semua order untuk menghentikan kerugian lebih lanjut.
5. **Penyegaran Portofolio**:
    - Pengguna mungkin ingin membatalkan semua order untuk memulai ulang atau merestrukturisasi strategi perdagangan mereka dari awal. Dengan **CancelAllOrders**, mereka dapat dengan mudah membersihkan slate dan mempersiapkan diri untuk menempatkan order baru yang sesuai dengan strategi baru.

Contoh Penggunaan:

- Seorang trader telah menempatkan beberapa order beli dan jual di pasar tetapi kemudian mendeteksi pergerakan pasar yang tidak sesuai dengan prediksi mereka. Untuk mencegah kerugian lebih lanjut, mereka memutuskan untuk membatalkan semua order terbuka dengan menggunakan **CancelAllOrders**, sehingga mereka dapat menilai kembali situasi sebelum menempatkan order baru.
- Dalam situasi pasar yang sangat volatile, seorang trader mungkin ingin segera menarik semua order mereka untuk menghindari eksekusi pada harga yang tidak diinginkan. Dengan **CancelAllOrders**, mereka dapat dengan cepat menghapus semua komitmen mereka di pasar.

8. CancelMultipleOrdersById

Deskripsi:

Instruksi **CancelMultipleOrdersById** memungkinkan pengguna untuk membatalkan beberapa order yang telah mereka tempatkan di order book berdasarkan ID order spesifik. Ini memberikan fleksibilitas yang lebih besar bagi pengguna untuk menargetkan dan membatalkan order tertentu tanpa harus membatalkan semua order yang mereka miliki. Fitur ini sangat berguna ketika pengguna ingin mengelola order mereka dengan lebih selektif, seperti dalam kasus di mana hanya sebagian dari order mereka yang tidak lagi relevan atau diinginkan.

Parameter Utama:

- **orders**:
    - Ini adalah daftar order yang ingin dibatalkan oleh pengguna. Setiap entri dalam daftar ini mencakup informasi berikut:
        - **side**: Menentukan apakah order yang dibatalkan adalah **Bid** (beli) atau **Ask** (jual). Ini membantu sistem mengidentifikasi sisi yang tepat di order book untuk mencari order yang akan dibatalkan.
        - **price_in_ticks**: Harga aset dalam bentuk ticks untuk order yang akan dibatalkan. Ini digunakan untuk menemukan order yang sesuai di order book.

- **orderSequenceNumber**: Nomor urut dari order yang akan dibatalkan. Ini adalah pengidentifikasi unik yang menunjukkan posisi order di dalam urutan order book, yang memungkinkan sistem untuk menargetkan order yang tepat untuk dibatalkan.

Fungsi Utama:

Instruksi **CancelMultipleOrdersById** berfungsi untuk memberikan pengguna kontrol yang lebih rinci dan selektif dalam membatalkan order mereka di order book. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pembatalan Selektif**:
   - **CancelMultipleOrdersById** memungkinkan pengguna untuk menargetkan dan membatalkan order tertentu yang tidak lagi relevan atau sesuai dengan strategi mereka. Ini memberi pengguna fleksibilitas untuk mempertahankan order lain yang masih valid atau diinginkan.
2. **Pengelolaan Risiko yang Lebih Baik**:
   - Dalam situasi di mana hanya sebagian dari order pengguna yang perlu dibatalkan, instruksi ini memungkinkan mereka untuk mengelola eksposur risiko mereka dengan lebih presisi, tanpa harus membatalkan semua order yang mereka miliki di pasar.
3. **Efisiensi dalam Manajemen Order**:
   - Daripada harus membatalkan order satu per satu secara manual, pengguna dapat menggunakan **CancelMultipleOrdersById** untuk membatalkan beberapa order sekaligus, menghemat waktu dan usaha dalam pengelolaan portofolio mereka.
4. **Penyesuaian Strategi Perdagangan**:
   - Ketika kondisi pasar berubah, pengguna mungkin ingin menyesuaikan strategi mereka dengan cepat. Dengan membatalkan hanya order tertentu, mereka dapat dengan cepat menyesuaikan posisi mereka tanpa gangguan besar pada keseluruhan strategi perdagangan mereka.
5. **Kontrol Penuh Terhadap Order Book**:
   - Instruksi ini memberikan pengguna kontrol penuh terhadap order mereka yang ada di order book, memungkinkan mereka untuk menargetkan dan mengelola order berdasarkan kondisi pasar yang dinamis dan kebutuhan perdagangan mereka.

Contoh Penggunaan:

- Seorang trader telah menempatkan beberapa order beli dan jual di pasar. Namun, setelah beberapa waktu, mereka memutuskan bahwa hanya beberapa order tertentu yang tidak lagi sesuai dengan strategi mereka. Dengan menggunakan **CancelMultipleOrdersById**, mereka dapat membatalkan order yang spesifik tersebut tanpa mempengaruhi order lain yang masih sesuai dengan tujuan mereka.
- Dalam skenario lain, seorang trader mungkin mendapati bahwa beberapa order yang mereka tempatkan tidak lagi relevan karena perubahan signifikan dalam harga pasar. Daripada membatalkan semua order, mereka dapat memilih untuk hanya membatalkan order yang terkait dengan harga yang tidak lagi realistis, menggunakan instruksi **CancelMultipleOrdersById**.

9. DecreaseOrder

Deskripsi:

Instruksi **DecreaseOrder** memungkinkan pengguna untuk mengurangi ukuran (quantity) dari order yang telah ditempatkan di order book. Ini berguna ketika pengguna ingin mengubah strategi perdagangan mereka tanpa harus membatalkan seluruh order yang sudah ada. Misalnya, jika pasar bergerak berlawanan dengan ekspektasi pengguna, mereka mungkin ingin mengurangi eksposur mereka dengan mengurangi jumlah order tanpa membatalkannya sepenuhnya.

Parameter Utama:

- **side**:
  - Menentukan apakah order yang dikurangi adalah **Bid** (order beli) atau **Ask** (order jual). Ini membantu sistem mengidentifikasi sisi yang tepat di order book untuk mengurangi ukuran order.
- **price_in_ticks**:
  - Harga order yang ingin dikurangi dalam bentuk ticks. Tick adalah unit terkecil dari harga dalam sistem. Sistem menggunakan harga ini untuk menemukan order yang sesuai di order book.
- **orderSequenceNumber**:
  - Nomor urut dari order yang akan dikurangi. Ini adalah pengidentifikasi unik yang menunjukkan posisi order di dalam urutan order book, yang memungkinkan sistem untuk menargetkan order yang tepat untuk dikurangi.
- **size**:
  - Jumlah lot dasar yang ingin dikurangi dari order yang ada. Ini menentukan seberapa besar pengurangan ukuran order tersebut.

Fungsi Utama:

Instruksi **DecreaseOrder** berfungsi untuk mengubah jumlah aset yang terkait dengan order yang telah ditempatkan di order book, memberikan fleksibilitas tambahan dalam manajemen order. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pengelolaan Risiko yang Lebih Baik**:
   - **DecreaseOrder** memungkinkan pengguna untuk mengurangi eksposur mereka terhadap risiko pasar dengan menurunkan jumlah order tanpa perlu membatalkan seluruh order. Ini penting dalam kondisi pasar yang tidak pasti atau ketika pengguna ingin menyesuaikan eksposur mereka secara bertahap.
2. **Fleksibilitas dalam Penyesuaian Order**:
   - Daripada membatalkan dan menempatkan kembali order baru, **DecreaseOrder** memungkinkan penyesuaian langsung terhadap order yang sudah ada, menghemat waktu dan potensi biaya terkait dengan menempatkan order baru.
3. **Menjaga Posisi di Order Book**:
   - Dengan menggunakan **DecreaseOrder**, pengguna dapat mengurangi ukuran order mereka tanpa kehilangan posisi di order book. Ini berarti order yang dikurangi tetap berada di posisi aslinya dalam antrian order, yang bisa sangat berharga dalam situasi di mana urutan eksekusi penting.
4. **Optimalisasi Likuiditas**:
   - **DecreaseOrder** dapat digunakan untuk menyesuaikan likuiditas yang tersedia di order book, memungkinkan pengguna untuk mengontrol seberapa banyak aset yang mereka tawarkan untuk dijual atau beli pada harga tertentu, sehingga menyeimbangkan strategi mereka dengan kondisi pasar yang ada.
5. **Efisiensi Pengelolaan Order**:

- Daripada harus membatalkan order yang besar dan menempatkan beberapa order yang lebih kecil, pengguna dapat dengan mudah mengurangi ukuran order yang ada sesuai kebutuhan mereka, memberikan efisiensi dalam manajemen portofolio dan pengambilan keputusan.

Contoh Penggunaan:

- Seorang trader menempatkan order beli yang besar di pasar dengan harapan harga akan turun. Namun, ketika harga tidak turun sebanyak yang diharapkan, trader tersebut memutuskan untuk mengurangi ukuran order beli mereka agar tidak membeli terlalu banyak aset pada harga yang lebih tinggi dari yang diinginkan. Dengan **DecreaseOrder**, mereka dapat mengurangi jumlah lot yang terkait dengan order tersebut tanpa membatalkannya sepenuhnya.
- Dalam skenario lain, seorang trader mungkin telah menempatkan order jual yang besar tetapi kemudian memutuskan bahwa mereka hanya ingin menjual sebagian dari aset tersebut. Mereka dapat menggunakan **DecreaseOrder** untuk mengurangi ukuran order jual mereka sambil tetap mempertahankan sisanya di order book.

10. DecreaseOrder

Deskripsi:

Instruksi **DecreaseOrder** memungkinkan pengguna untuk mengurangi ukuran (quantity) dari order yang telah ditempatkan di order book. Ini berguna ketika pengguna ingin mengubah strategi perdagangan mereka tanpa harus membatalkan seluruh order yang sudah ada. Misalnya, jika pasar bergerak berlawanan dengan ekspektasi pengguna, mereka mungkin ingin mengurangi eksposur mereka dengan mengurangi jumlah order tanpa membatalkannya sepenuhnya.

Parameter Utama:

- **side**:
  - Menentukan apakah order yang dikurangi adalah **Bid** (order beli) atau **Ask** (order jual). Ini membantu sistem mengidentifikasi sisi yang tepat di order book untuk mengurangi ukuran order.
- **price_in_ticks**:
  - Harga order yang ingin dikurangi dalam bentuk ticks. Tick adalah unit terkecil dari harga dalam sistem. Sistem menggunakan harga ini untuk menemukan order yang sesuai di order book.
- **orderSequenceNumber**:
  - Nomor urut dari order yang akan dikurangi. Ini adalah pengidentifikasi unik yang menunjukkan posisi order di dalam urutan order book, yang memungkinkan sistem untuk menargetkan order yang tepat untuk dikurangi.
- **size**:
  - Jumlah lot dasar yang ingin dikurangi dari order yang ada. Ini menentukan seberapa besar pengurangan ukuran order tersebut.

Fungsi Utama:

Instruksi **DecreaseOrder** berfungsi untuk mengubah jumlah aset yang terkait dengan order yang telah ditempatkan di order book, memberikan fleksibilitas tambahan dalam manajemen order. Berikut adalah

beberapa fungsi utama dari instruksi ini:

1. **Pengelolaan Risiko yang Lebih Baik**:
   - **DecreaseOrder** memungkinkan pengguna untuk mengurangi eksposur mereka terhadap risiko pasar dengan menurunkan jumlah order tanpa perlu membatalkan seluruh order. Ini penting dalam kondisi pasar yang tidak pasti atau ketika pengguna ingin menyesuaikan eksposur mereka secara bertahap.
2. **Fleksibilitas dalam Penyesuaian Order**:
   - Daripada membatalkan dan menempatkan kembali order baru, **DecreaseOrder** memungkinkan penyesuaian langsung terhadap order yang sudah ada, menghemat waktu dan potensi biaya terkait dengan menempatkan order baru.
3. **Menjaga Posisi di Order Book**:
   - Dengan menggunakan **DecreaseOrder**, pengguna dapat mengurangi ukuran order mereka tanpa kehilangan posisi di order book. Ini berarti order yang dikurangi tetap berada di posisi aslinya dalam antrian order, yang bisa sangat berharga dalam situasi di mana urutan eksekusi penting.
4. **Optimalisasi Likuiditas**:
   - **DecreaseOrder** dapat digunakan untuk menyesuaikan likuiditas yang tersedia di order book, memungkinkan pengguna untuk mengontrol seberapa banyak aset yang mereka tawarkan untuk dijual atau beli pada harga tertentu, sehingga menyeimbangkan strategi mereka dengan kondisi pasar yang ada.
5. **Efisiensi Pengelolaan Order**:
   - Daripada harus membatalkan order yang besar dan menempatkan beberapa order yang lebih kecil, pengguna dapat dengan mudah mengurangi ukuran order yang ada sesuai kebutuhan mereka, memberikan efisiensi dalam manajemen portofolio dan pengambilan keputusan.

Contoh Penggunaan:

- Seorang trader menempatkan order beli yang besar di pasar dengan harapan harga akan turun. Namun, ketika harga tidak turun sebanyak yang diharapkan, trader tersebut memutuskan untuk mengurangi ukuran order beli mereka agar tidak membeli terlalu banyak aset pada harga yang lebih tinggi dari yang diinginkan. Dengan **DecreaseOrder**, mereka dapat mengurangi jumlah lot yang terkait dengan order tersebut tanpa membatalkannya sepenuhnya.
- Dalam skenario lain, seorang trader mungkin telah menempatkan order jual yang besar tetapi kemudian memutuskan bahwa mereka hanya ingin menjual sebagian dari aset tersebut. Mereka dapat menggunakan **DecreaseOrder** untuk mengurangi ukuran order jual mereka sambil tetap mempertahankan sisanya di order book.

11. ForceCancelOrders

Deskripsi:

Instruksi **ForceCancelOrders** memungkinkan otoritas pasar atau entitas yang berwenang untuk secara paksa membatalkan satu atau lebih order yang ada di order book dalam pasar tertentu. Instruksi ini digunakan dalam situasi luar biasa di mana perlu dilakukan tindakan cepat untuk melindungi integritas pasar, menghentikan aktivitas perdagangan yang tidak diinginkan, atau merespons perubahan mendadak

dalam kondisi pasar. Dengan instruksi ini, pengelola pasar dapat menghapus order yang mungkin merugikan pasar atau pengguna lain, atau mengelola risiko dengan cara yang lebih efektif.

Parameter Utama:

- **market**:
  - Pasar di mana order akan dibatalkan. Ini mengidentifikasi pasar spesifik di mana order yang ada di order book akan dihapus.
- **side**:
  - Menentukan apakah order yang akan dibatalkan adalah **Bid** (order beli) atau **Ask** (order jual). Ini membantu sistem mengidentifikasi sisi order book yang relevan.
- **tickLimit** (opsional):
  - Batas harga dalam tick untuk membatalkan order. Ini memungkinkan pembatalan order yang berada di rentang harga tertentu, memberikan kontrol lebih terhadap bagaimana order dibatalkan.
- **numOrdersToSearch** (opsional):
  - Jumlah maksimum order yang akan dicari dalam order book untuk dibatalkan. Ini mengontrol seberapa luas pencarian order yang akan dibatalkan.
- **numOrdersToCancel** (opsional):
  - Jumlah maksimum order yang akan dibatalkan. Pengelola pasar dapat menentukan berapa banyak order yang akan dibatalkan, memungkinkan kontrol yang lebih presisi atas tindakan ini.

Fungsi Utama:

Instruksi **ForceCancelOrders** berfungsi untuk memberikan otoritas pasar kemampuan untuk membatalkan order secara paksa di order book, yang bisa dilakukan dalam kondisi-kondisi darurat atau ketika tindakan cepat diperlukan untuk menjaga stabilitas pasar. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Manajemen Risiko yang Cepat**:
   - **ForceCancelOrders** memberikan pengelola pasar kemampuan untuk segera menghapus order yang berpotensi merugikan atau yang dapat mengganggu keseimbangan pasar. Ini sangat penting dalam situasi di mana pasar menjadi sangat volatil atau ketika aktivitas perdagangan tidak diinginkan terdeteksi.
2. **Perlindungan Terhadap Manipulasi Pasar**:
   - Instruksi ini dapat digunakan untuk membatalkan order yang dianggap sebagai bagian dari aktivitas manipulasi pasar, melindungi integritas pasar dan mencegah kerugian yang tidak diinginkan bagi pengguna lain.
3. **Penghentian Aktivitas Perdagangan Tertentu**:
   - Pengelola pasar dapat menggunakan **ForceCancelOrders** untuk menghentikan aktivitas perdagangan tertentu yang mungkin telah menjadi masalah, seperti order yang salah tempat atau order yang tidak lagi relevan dengan kondisi pasar saat ini.
4. **Fleksibilitas dalam Pembatalan Order**:
   - Dengan parameter seperti **tickLimit**, **numOrdersToSearch**, dan **numOrdersToCancel**, instruksi ini memberikan pengelola pasar kontrol penuh atas proses pembatalan, memungkinkan tindakan yang disesuaikan dengan kebutuhan spesifik pasar.
5. **Reaksi Terhadap Kondisi Darurat**:

- **ForceCancelOrders** dapat digunakan dalam situasi darurat di mana tindakan cepat diperlukan untuk mencegah kerugian besar atau untuk menjaga kestabilan pasar. Ini memungkinkan pasar untuk tetap fungsional bahkan dalam kondisi yang sangat menantang.

Contoh Penggunaan:

- Dalam situasi di mana pasar mengalami lonjakan harga yang tidak biasa karena order yang salah tempat atau salah ketik, pengelola pasar dapat menggunakan **ForceCancelOrders** untuk membatalkan order-order ini dan mengembalikan pasar ke keadaan normal.
- Jika ditemukan bahwa ada aktivitas perdagangan yang mencurigakan atau manipulatif di pasar, pengelola dapat secara proaktif membatalkan order yang terkait menggunakan instruksi ini, sehingga mencegah aktivitas tersebut dari merugikan peserta lain di pasar.
- Dalam kondisi pasar yang sangat volatil, di mana harga aset bergerak dengan cepat dan dapat menyebabkan kerugian signifikan, pengelola pasar dapat menggunakan **ForceCancelOrders** untuk membatalkan order yang sudah tidak relevan atau yang bisa berpotensi merusak keseimbangan pasar.

12. ApplyForSeat

Deskripsi:

Instruksi **ApplyForSeat** memungkinkan pengguna untuk mengajukan permohonan untuk mendapatkan "seat" atau posisi di pasar tertentu dalam platform **Recto Verso**. Dalam konteks platform ini, memiliki seat berarti pengguna dapat berpartisipasi secara aktif dalam pasar tersebut, termasuk menempatkan order, berinteraksi dengan order book, dan mungkin mendapatkan manfaat tertentu yang hanya tersedia bagi pemegang seat. Instruksi ini biasanya digunakan oleh pengguna yang ingin menjadi peserta aktif di pasar tertentu dan berpartisipasi dalam aktivitas perdagangan dengan hak akses yang telah ditetapkan.

Parameter Utama:

- **market**:
  - Pasar di mana pengguna ingin mengajukan permohonan untuk seat. Ini mengidentifikasi pasar spesifik di mana pengguna ingin berpartisipasi.
- **seat_authority**:
  - Otoritas yang berwenang untuk menyetujui atau menolak permohonan seat. Ini biasanya merupakan entitas atau akun yang memiliki kendali atas alokasi seat di pasar tersebut.
- **trader** (opsional):
  - Alamat akun pengguna yang mengajukan permohonan seat. Jika tidak ditentukan, biasanya mengacu pada akun yang mengirimkan instruksi ini.

Fungsi Utama:

Instruksi **ApplyForSeat** berfungsi untuk mendaftarkan pengguna sebagai peserta yang berwenang di pasar tertentu, memberikan mereka akses dan hak untuk berpartisipasi dalam perdagangan di pasar tersebut. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Partisipasi dalam Pasar**:

- **ApplyForSeat** memungkinkan pengguna untuk secara resmi menjadi peserta di pasar tertentu. Dengan memiliki seat, pengguna dapat menempatkan order, berinteraksi dengan order book, dan memanfaatkan fasilitas perdagangan yang tersedia di pasar tersebut.

2. **Kontrol Akses**:
   - Dengan adanya parameter **seat_authority**, instruksi ini memastikan bahwa hanya pengguna yang disetujui oleh otoritas yang berwenang yang dapat memiliki seat di pasar. Ini memberikan kontrol yang lebih besar atas siapa yang dapat berpartisipasi dalam pasar, menjaga integritas dan keamanan pasar.
3. **Alokasi Seat yang Efisien**:
   - Instruksi ini memungkinkan otoritas pasar untuk dengan mudah mengelola alokasi seat kepada pengguna yang berminat. Otoritas dapat menyetujui atau menolak permohonan berdasarkan kebijakan mereka, memastikan bahwa seat diberikan kepada pengguna yang dianggap sesuai.
4. **Manfaat Eksklusif**:
   - Memiliki seat mungkin memberikan pengguna akses ke manfaat atau fitur eksklusif yang tidak tersedia bagi non-pemegang seat. Ini bisa termasuk prioritas dalam eksekusi order, biaya transaksi yang lebih rendah, atau hak suara dalam keputusan pasar.
5. **Peningkatan Likuiditas Pasar**:
   - Dengan mendorong pengguna untuk memiliki seat, platform **Recto Verso** dapat meningkatkan likuiditas di pasar, karena lebih banyak pengguna yang aktif dalam menempatkan order dan berinteraksi dengan order book.

Contoh Penggunaan:

- Seorang trader ingin berpartisipasi dalam pasar baru yang baru saja diluncurkan di platform **Recto Verso**. Untuk mendapatkan akses penuh ke pasar tersebut, mereka menggunakan instruksi **ApplyForSeat** untuk mengajukan permohonan seat. Setelah permohonan mereka disetujui oleh **seat_authority**, mereka dapat mulai menempatkan order dan berdagang di pasar tersebut.
- Dalam skenario lain, sebuah entitas institusional yang ingin berpartisipasi dalam volume perdagangan besar di pasar tertentu dapat menggunakan **ApplyForSeat** untuk memastikan mereka memiliki seat yang memberikan mereka hak akses prioritas dalam perdagangan di pasar tersebut.

13. ReserveSeatAuthorized

Deskripsi:

Instruksi **ReserveSeatAuthorized** memungkinkan otoritas pasar atau entitas yang berwenang untuk mereservasi atau memesan seat (posisi) di pasar tertentu bagi seorang pengguna yang ditunjuk. Ini adalah alat penting bagi pengelola pasar yang ingin memastikan bahwa seat tertentu dialokasikan untuk individu atau entitas yang telah disetujui, sebelum mereka secara resmi mengajukan permohonan atau sebelum pasar dibuka untuk umum. Instruksi ini membantu menjaga kontrol atas alokasi seat dan memastikan bahwa seat tersedia untuk peserta yang diinginkan tanpa harus melalui proses aplikasi standar.

Parameter Utama:

- **market**:

    - Pasar di mana seat akan direreservasi. Ini mengidentifikasi pasar spesifik yang seat-nya akan dialokasikan kepada pengguna yang ditunjuk.
- **seat**:
    - Seat spesifik yang akan direreservasi untuk pengguna. Ini adalah pengenal unik untuk seat yang akan dialokasikan.
- **user**:
    - Alamat akun pengguna yang akan menerima seat yang direreservasi. Ini adalah pengguna yang telah dipilih untuk mendapatkan seat oleh otoritas pasar.

Fungsi Utama:

Instruksi **ReserveSeatAuthorized** berfungsi untuk memastikan bahwa seat tertentu dialokasikan kepada pengguna yang ditunjuk oleh otoritas pasar, memberikan kontrol yang lebih besar atas proses alokasi seat dan menjaga integritas pasar. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Kontrol Penuh atas Alokasi Seat**:
    - **ReserveSeatAuthorized** memungkinkan otoritas pasar untuk memastikan bahwa seat tertentu disediakan untuk pengguna yang telah dipilih. Ini bisa berguna dalam situasi di mana seat terbatas atau diinginkan untuk dialokasikan kepada peserta tertentu.
2. **Memastikan Partisipasi Pengguna yang Diinginkan**:
    - Instruksi ini memastikan bahwa seat dialokasikan kepada individu atau entitas yang telah diidentifikasi oleh otoritas pasar sebagai peserta yang diinginkan atau penting. Hal ini bisa dilakukan sebelum pasar dibuka secara luas untuk peserta lainnya.
3. **Penyederhanaan Proses Alokasi**:
    - Dengan mereservasi seat untuk pengguna tertentu, otoritas pasar dapat menghindari proses aplikasi yang lebih rumit dan langsung mengalokasikan seat kepada pengguna yang telah memenuhi syarat atau telah disetujui sebelumnya.
4. **Fleksibilitas dalam Pengelolaan Pasar**:
    - Instruksi ini memberikan fleksibilitas bagi pengelola pasar untuk mengelola siapa yang dapat berpartisipasi dalam pasar mereka, dan memastikan bahwa seat tidak diberikan secara sembarangan tetapi dialokasikan dengan strategi tertentu.
5. **Manajemen Seat yang Proaktif**:
    - Dengan **ReserveSeatAuthorized**, pengelola pasar dapat proaktif dalam mengelola alokasi seat, memastikan bahwa seat tersedia untuk pengguna yang penting atau strategis, sebelum pengguna lain dapat mengklaimnya.

Contoh Penggunaan:

- Sebuah pasar baru akan segera diluncurkan, dan otoritas pasar ingin memastikan bahwa beberapa seat pertama dialokasikan kepada mitra strategis atau pengguna VIP. Dengan menggunakan **ReserveSeatAuthorized**, mereka dapat mereservasi seat-seats ini untuk akun-akun yang telah ditentukan, memastikan bahwa mereka mendapatkan akses ke pasar tersebut.
- Dalam situasi lain, sebuah entitas yang akan berpartisipasi secara signifikan dalam volume perdagangan pasar mungkin perlu dipastikan memiliki seat yang sesuai sebelum pasar dibuka untuk umum. Otoritas pasar dapat menggunakan **ReserveSeatAuthorized** untuk mereservasi seat untuk entitas ini, memberikan mereka akses prioritas.

14. ModifySeatState

Deskripsi:

Instruksi **ModifySeatState** memungkinkan otoritas pasar atau entitas yang berwenang untuk mengubah status dari "seat" yang dimiliki oleh peserta di dalam pasar tertentu di platform **Recto Verso**. Seat dalam konteks ini mengacu pada hak atau posisi yang dimiliki oleh seorang peserta di pasar yang memberi mereka akses untuk berpartisipasi dalam aktivitas perdagangan. Instruksi ini memberikan fleksibilitas kepada pengelola pasar untuk mengatur status partisipasi pengguna, baik itu menyetujui, menolak, atau mencabut akses mereka berdasarkan kriteria yang ditetapkan oleh pasar.

Parameter Utama:

- **market**:
  - Pasar yang seat-nya akan dimodifikasi. Ini mengidentifikasi pasar spesifik di mana status seat peserta akan diubah.
- **seat**:
  - Seat yang akan dimodifikasi statusnya. Ini adalah pengenal unik untuk seat yang terdaftar di pasar tersebut.
- **approvalStatus**:
  - Status baru yang akan diterapkan pada seat. Status yang mungkin termasuk:
    - **NotApproved**: Seat belum disetujui untuk berpartisipasi dalam pasar. Ini bisa berarti bahwa permohonan seat masih dalam proses atau telah ditolak.
    - **Approved**: Seat telah disetujui dan peserta dapat secara aktif berpartisipasi dalam perdagangan di pasar tersebut.
    - **Retired**: Seat tidak lagi aktif atau telah dicabut, dan peserta tidak lagi memiliki akses untuk berpartisipasi dalam aktivitas pasar.

Fungsi Utama:

Instruksi **ModifySeatState** berfungsi untuk mengelola status keanggotaan atau partisipasi pengguna di pasar, memberikan pengelola pasar alat untuk mengatur siapa yang dapat atau tidak dapat berpartisipasi dalam aktivitas perdagangan. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Manajemen Partisipasi Peserta**:
   - **ModifySeatState** memungkinkan pengelola pasar untuk mengatur siapa yang diizinkan untuk berpartisipasi dalam perdagangan di pasar tertentu. Ini mencakup persetujuan seat baru, penolakan permohonan, atau pencabutan seat yang sudah ada.
2. **Kontrol Akses yang Lebih Baik**:
   - Dengan instruksi ini, pengelola pasar dapat memastikan bahwa hanya peserta yang memenuhi kriteria tertentu yang dapat berpartisipasi di pasar. Ini bisa menjadi bagian dari strategi untuk menjaga kualitas pasar atau menghindari risiko yang ditimbulkan oleh peserta yang tidak diinginkan.
3. **Fleksibilitas dalam Pengelolaan Peserta**:

    - Pengelola pasar dapat dengan mudah mengubah status seat berdasarkan perubahan kondisi atau kebutuhan pasar. Misalnya, seat yang awalnya disetujui dapat dicabut jika peserta melanggar aturan pasar atau jika kondisi berubah.
4. **Perlindungan Terhadap Pasar**:
    - Instruksi ini juga memungkinkan pasar untuk melindungi diri dari aktivitas yang tidak diinginkan dengan mengubah status seat pengguna yang berpotensi merugikan pasar atau yang tidak lagi memenuhi kriteria partisipasi.
5. **Pengelolaan Dinamis Seat**:
    - **ModifySeatState** memberikan kemampuan untuk mengelola seat secara dinamis, memungkinkan pengelola pasar untuk merespons dengan cepat terhadap kebutuhan atau situasi yang berubah, seperti ketika peserta baru ingin bergabung atau ketika seat harus dicabut untuk menjaga stabilitas pasar.

Contoh Penggunaan:

- Seorang pengguna baru yang mengajukan permohonan seat di pasar mungkin awalnya berada dalam status **NotApproved**. Setelah pengelola pasar memverifikasi informasi dan memutuskan untuk menerima pengguna tersebut, mereka dapat menggunakan **ModifySeatState** untuk mengubah status seat menjadi **Approved**, sehingga pengguna tersebut dapat mulai berpartisipasi dalam perdagangan.
- Jika seorang peserta yang telah disetujui untuk seat melanggar aturan pasar atau terlibat dalam aktivitas yang mencurigakan, pengelola pasar dapat menggunakan **ModifySeatState** untuk mengubah status seat menjadi **Retired**, yang secara efektif mencabut hak mereka untuk berpartisipasi di pasar tersebut.
- Dalam skenario lain, sebuah pasar yang ingin memperbarui kriteria partisipasi mungkin perlu mengubah status beberapa seat dari **Approved** menjadi **NotApproved** hingga peserta memenuhi persyaratan baru. Ini bisa dilakukan dengan mudah menggunakan instruksi ini.

15. RemoveSeat

Deskripsi:

Instruksi **RemoveSeat** memungkinkan otoritas pasar atau entitas yang berwenang untuk mencabut atau menghapus seat (posisi) yang telah diberikan kepada seorang pengguna di pasar tertentu di platform **Recto Verso**. Dengan instruksi ini, pengelola pasar dapat mengatur ulang akses partisipasi di pasar dengan menghilangkan seat dari pengguna yang tidak lagi memenuhi syarat, melanggar aturan, atau tidak diperlukan lagi. **RemoveSeat** adalah bagian penting dari manajemen pasar yang memastikan bahwa hanya peserta yang memenuhi kriteria yang dapat tetap berpartisipasi dalam aktivitas perdagangan.

Parameter Utama:

- **market**:
    - Pasar di mana seat akan dihapus. Ini mengidentifikasi pasar spesifik yang seat-nya akan dicabut dari pengguna.
- **seat**:

    - Seat yang akan dihapus. Ini adalah pengenal unik untuk seat yang telah diberikan kepada pengguna di pasar tersebut.
- **user**:
    - Alamat akun pengguna yang seat-nya akan dihapus. Ini adalah peserta yang tidak lagi diizinkan untuk berpartisipasi dalam aktivitas pasar.

Fungsi Utama:

Instruksi **RemoveSeat** berfungsi untuk mencabut seat yang telah dialokasikan kepada pengguna di pasar, memberikan pengelola pasar kemampuan untuk mengelola akses partisipasi secara dinamis dan memastikan bahwa hanya pengguna yang memenuhi syarat yang dapat berpartisipasi. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pengelolaan Akses Partisipasi**:
    - **RemoveSeat** memungkinkan pengelola pasar untuk mengelola akses partisipasi dengan mencabut seat dari pengguna yang tidak lagi memenuhi kriteria atau yang melanggar aturan pasar. Ini menjaga integritas pasar dan memastikan bahwa hanya peserta yang memenuhi syarat yang dapat berdagang.
2. **Perlindungan Pasar**:
    - Dengan mencabut seat dari pengguna yang tidak diinginkan atau yang berpotensi merugikan, instruksi ini membantu melindungi pasar dari aktivitas yang tidak diinginkan atau manipulatif. Ini juga dapat digunakan untuk mengurangi risiko yang ditimbulkan oleh peserta yang tidak sesuai.
3. **Pemulihan Likuiditas atau Sumber Daya**:
    - Menghapus seat dari pengguna dapat membantu dalam pemulihan likuiditas atau sumber daya pasar yang mungkin telah dialokasikan untuk pengguna tersebut. Ini memungkinkan pengelola pasar untuk mengalokasikan ulang seat kepada peserta lain yang lebih cocok atau lebih sesuai dengan tujuan pasar.
4. **Penyelarasan dengan Kebijakan Pasar**:
    - Instruksi ini memungkinkan pengelola pasar untuk memastikan bahwa semua peserta tetap mematuhi kebijakan pasar. Jika ada perubahan dalam kebijakan atau aturan pasar, seat dapat dicabut dari pengguna yang tidak mematuhi atau tidak sesuai dengan kriteria baru.
5. **Manajemen Peserta yang Dinamis**:
    - **RemoveSeat** memberikan fleksibilitas kepada pengelola pasar untuk secara dinamis mengelola peserta, memastikan bahwa pasar tetap sehat, kompetitif, dan bebas dari aktivitas yang tidak diinginkan.

Contoh Penggunaan:

- Seorang peserta yang memiliki seat di pasar telah melanggar aturan pasar dengan melakukan aktivitas perdagangan yang manipulatif. Pengelola pasar dapat menggunakan **RemoveSeat** untuk mencabut seat mereka, sehingga mereka tidak lagi dapat berpartisipasi dalam aktivitas perdagangan di pasar tersebut.
- Sebuah entitas yang sebelumnya diizinkan untuk memiliki seat di pasar mengalami perubahan status atau tidak lagi memenuhi kriteria partisipasi. Otoritas pasar dapat menggunakan **RemoveSeat** untuk mencabut seat mereka dan mungkin mengalokasikan seat tersebut kepada peserta lain yang lebih sesuai.

16. AppointSuccessor

Deskripsi:

Instruksi **AppointSuccessor** memungkinkan pemegang otoritas pasar saat ini untuk menunjuk penerus baru yang akan mengambil alih otoritas tersebut di masa mendatang. Ini adalah bagian penting dari pengelolaan pasar yang berkelanjutan, memastikan bahwa kendali atas pasar dapat dialihkan dengan lancar dan sesuai dengan rencana suksesi yang telah ditetapkan. Instruksi ini memberikan jaminan bahwa pasar atau entitas di dalam **Recto Verso** akan terus dikelola secara efektif, bahkan jika pemegang otoritas saat ini memutuskan untuk mengalihkan tanggung jawab mereka atau jika terjadi keadaan darurat yang memerlukan penggantian otoritas.

Parameter Utama:

- **market**:
  - Pasar di mana penerus baru akan diangkat. Ini mengidentifikasi pasar spesifik yang akan memiliki otoritas baru setelah penerus diangkat.
- **successor**:
  - Alamat akun penerus yang ditunjuk untuk mengambil alih otoritas. Ini adalah pengguna atau entitas yang akan ditunjuk sebagai pemegang otoritas berikutnya.
- **current_authority** (opsional):
  - Alamat akun otoritas saat ini. Parameter ini dapat digunakan untuk memastikan bahwa instruksi hanya dapat dijalankan oleh otoritas yang sah.

Fungsi Utama:

Instruksi **AppointSuccessor** berfungsi untuk menetapkan penerus baru untuk otoritas pasar, memberikan jaminan bahwa otoritas pasar dapat dialihkan dengan cara yang terstruktur dan terencana. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pengelolaan Suksesi yang Terencana**:
   - **AppointSuccessor** memungkinkan otoritas pasar saat ini untuk menetapkan penerus yang akan mengambil alih tanggung jawab di masa depan. Ini membantu memastikan bahwa transisi otoritas berjalan dengan lancar dan sesuai dengan strategi suksesi yang telah ditetapkan.
2. **Kontinuitas Operasional**:
   - Dengan menetapkan penerus melalui instruksi ini, platform **Recto Verso** memastikan bahwa pasar dapat terus beroperasi tanpa gangguan, meskipun terjadi perubahan dalam kepemimpinan atau otoritas pasar. Ini sangat penting untuk menjaga kepercayaan pengguna dan stabilitas pasar.
3. **Keamanan dan Verifikasi**:
   - Parameter opsional seperti **current_authority** memberikan lapisan keamanan tambahan, memastikan bahwa hanya otoritas sah yang dapat menunjuk penerus. Ini membantu mencegah pengambilalihan yang tidak sah atau tidak diinginkan.
4. **Fleksibilitas dalam Manajemen Otoritas**:
   - **AppointSuccessor** memberikan fleksibilitas bagi pemegang otoritas untuk merencanakan masa depan pasar atau entitas yang mereka kelola. Mereka dapat memilih penerus yang mereka

percayai untuk melanjutkan visi dan strategi mereka.

5. **Peningkatan Keberlanjutan Pasar**:
    - Instruksi ini membantu meningkatkan keberlanjutan dan stabilitas pasar dengan memastikan bahwa ada penerus yang siap mengambil alih jika diperlukan. Ini juga memberi pengguna dan investor kepercayaan bahwa pasar akan dikelola secara konsisten di masa depan.

Contoh Penggunaan:

- Seorang pemegang otoritas pasar yang berencana untuk pensiun atau beralih ke proyek lain dapat menggunakan **AppointSuccessor** untuk menunjuk penerus yang akan mengambil alih tanggung jawab mereka. Penerus ini kemudian akan memiliki otoritas untuk mengelola pasar sesuai dengan kebijakan yang telah ditetapkan.
- Dalam situasi di mana sebuah organisasi mengelola beberapa pasar dan ingin memastikan bahwa setiap pasar memiliki penerus yang jelas, mereka dapat menggunakan **AppointSuccessor** untuk menetapkan penerus di setiap pasar. Ini membantu memastikan bahwa jika ada perubahan dalam kepemimpinan, pasar tetap berjalan dengan lancar.

17. ClaimAuthority

Deskripsi:

Instruksi **ClaimAuthority** memungkinkan pengguna untuk mengklaim otoritas atas pasar tertentu atau entitas dalam platform **Recto Verso**. Otoritas ini mungkin telah ditransfer, diwariskan, atau dialihkan dari entitas lain, dan dengan instruksi ini, pengguna dapat mengambil kendali atas fungsi-fungsi pasar yang membutuhkan otoritas khusus, seperti mengelola parameter pasar, mengumpulkan fee, atau mengatur kebijakan perdagangan. **ClaimAuthority** adalah bagian penting dari manajemen pasar yang desentralisasi, memungkinkan pengguna untuk memastikan bahwa otoritas dipegang oleh pihak yang benar-benar memiliki hak tersebut.

Parameter Utama:

- **market**:
    - Pasar di mana otoritas akan diklaim. Ini mengidentifikasi pasar spesifik yang otoritasnya akan diambil alih oleh pengguna yang memanggil instruksi ini.
- **new_authority**:
    - Alamat akun baru yang akan menjadi otoritas setelah klaim dilakukan. Ini adalah pengguna atau entitas yang akan mengendalikan pasar setelah instruksi dijalankan.
- **previous_authority** (opsional):
    - Alamat akun otoritas sebelumnya. Meskipun ini mungkin tidak selalu diperlukan, beberapa implementasi mungkin menggunakan parameter ini untuk memastikan bahwa klaim dilakukan oleh pihak yang benar-benar memiliki hak untuk mengalihkan otoritas.

Fungsi Utama:

Instruksi **ClaimAuthority** berfungsi untuk memindahkan atau mengklaim otoritas atas pasar atau entitas tertentu di platform **Recto Verso**, memastikan bahwa kontrol dan manajemen pasar berada di tangan

yang benar. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pengalihan Otoritas yang Sah**:
   - **ClaimAuthority** memastikan bahwa otoritas atas pasar atau entitas tertentu dapat dialihkan secara sah dan transparan. Ini memungkinkan proses pengalihan kekuasaan yang lancar antara pengguna, baik sebagai bagian dari perencanaan suksesi atau pengalihan tanggung jawab.
2. **Pengelolaan Pasar yang Terdesentralisasi**:
   - Dalam sistem desentralisasi seperti **Recto Verso**, instruksi ini memastikan bahwa pengelolaan pasar tetap berada di tangan komunitas atau entitas yang telah ditunjuk, mengurangi risiko sentralisasi dan menjaga integritas pasar.
3. **Fleksibilitas dalam Manajemen Pasar**:
   - Instruksi ini memberikan fleksibilitas bagi pengguna untuk mengelola pasar atau entitas lain dengan lebih efektif, memungkinkan transfer otoritas sesuai kebutuhan operasional atau strategi bisnis.
4. **Keamanan dan Verifikasi**:
   - Dengan parameter opsional seperti **previous_authority**, sistem dapat memastikan bahwa hanya pengguna yang sah yang dapat mengklaim otoritas, menambahkan lapisan keamanan tambahan untuk mencegah pengambilalihan yang tidak sah.
5. **Kontinuitas dalam Operasi Pasar**:
   - **ClaimAuthority** membantu memastikan bahwa operasi pasar dapat berlanjut tanpa gangguan meskipun ada perubahan dalam otoritas, karena proses klaim yang cepat dan jelas.

Contoh Penggunaan:

- Seorang administrator yang awalnya menciptakan pasar di **Recto Verso** memutuskan untuk mengalihkan tanggung jawab pengelolaan pasar kepada anggota tim lain atau entitas baru. Dengan menggunakan **ClaimAuthority**, anggota tim baru tersebut dapat secara resmi mengambil alih otoritas atas pasar, memungkinkan mereka untuk mengelola parameter pasar, mengumpulkan fee, dan melaksanakan fungsi lainnya yang terkait dengan otoritas pasar.

18. CollectFees

Deskripsi:

Instruksi **CollectFees** memungkinkan otoritas pasar atau entitas yang berwenang untuk mengumpulkan fee (biaya) yang telah terakumulasi dari aktivitas perdagangan di pasar tertentu di platform **Recto Verso**. Setiap transaksi yang terjadi di pasar, seperti eksekusi order, biasanya dikenakan fee yang dibayarkan oleh pengguna. Fee ini dikumpulkan dan disimpan di akun khusus yang terkait dengan pasar tersebut. Melalui instruksi **CollectFees**, otoritas pasar dapat memindahkan fee yang terkumpul dari akun pasar ke akun penerima yang telah ditentukan, yang bisa berupa akun pribadi atau entitas yang mengelola pasar.

Parameter Utama:

- **market**:
   - Pasar dari mana fee akan dikumpulkan. Ini mengidentifikasi pasar spesifik di mana fee telah terakumulasi dari aktivitas perdagangan.

- **recipient**:
  - Alamat akun yang akan menerima fee yang dikumpulkan. Ini adalah pengguna atau entitas yang telah ditunjuk untuk menerima fee yang terkumpul di pasar tersebut.
- **feeCollector**:
  - Alamat akun yang memiliki hak untuk mengeksekusi instruksi **CollectFees**. Biasanya ini adalah otoritas pasar atau entitas yang berwenang yang bertanggung jawab atas pengelolaan pasar.

Fungsi Utama:

Instruksi **CollectFees** berfungsi untuk memindahkan fee yang telah terkumpul dari aktivitas perdagangan di pasar ke akun yang telah ditentukan. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pemindahan Fee yang Efisien**:
   - **CollectFees** memungkinkan pengelola pasar untuk secara efisien mengumpulkan dan memindahkan fee yang terkumpul dari akun pasar ke akun penerima. Ini adalah bagian penting dari proses pengelolaan pendapatan bagi pasar.
2. **Peningkatan Transparansi**:
   - Dengan instruksi ini, ada transparansi yang lebih besar dalam pengelolaan fee, karena semua fee yang dikumpulkan dapat dilacak dan dipindahkan dengan cara yang terstruktur dan terdokumentasi.
3. **Pemberian Imbalan kepada Pengelola Pasar**:
   - Fee yang dikumpulkan dapat digunakan sebagai imbalan bagi pengelola pasar atau entitas lain yang mendukung operasi pasar. Ini memberikan insentif yang diperlukan untuk menjaga dan meningkatkan kualitas pasar.
4. **Pengelolaan Dana Pasar**:
   - Instruksi ini memungkinkan pengelola pasar untuk secara teratur mengumpulkan fee yang terkumpul dan memindahkannya ke akun yang aman dan dapat diakses, membantu dalam manajemen dana yang lebih baik.
5. **Fleksibilitas dalam Penetapan Penerima**:
   - Dengan parameter **recipient**, pengelola pasar memiliki fleksibilitas untuk menentukan siapa yang akan menerima fee yang terkumpul, yang bisa berubah sesuai dengan kebijakan pasar atau persetujuan bersama.

Contoh Penggunaan:

- Sebuah pasar di platform **Recto Verso** telah beroperasi selama beberapa waktu, dan fee dari berbagai transaksi telah terkumpul di akun pasar. Pengelola pasar, yang memiliki otoritas sebagai **feeCollector**, menggunakan instruksi **CollectFees** untuk memindahkan fee yang terkumpul ke akun pribadi mereka atau ke akun entitas yang mengelola pasar.
- Dalam skenario lain, fee yang terkumpul selama periode waktu tertentu digunakan untuk mendanai pengembangan lebih lanjut dari pasar atau proyek terkait lainnya. Pengelola pasar mengumpulkan fee menggunakan **CollectFees** dan mengalokasikannya sesuai dengan rencana keuangan yang telah disetujui.

19. ModifyFeeRecipient

Deskripsi:

Instruksi **ModifyFeeRecipient** memungkinkan otoritas pasar atau entitas yang berwenang untuk mengubah penerima fee (biaya) yang telah ditetapkan dalam pasar tertentu di platform **Recto Verso**. Setiap kali transaksi terjadi di pasar, fee yang dihasilkan dari aktivitas perdagangan biasanya dialokasikan ke akun penerima yang telah ditentukan sebelumnya. Namun, dalam beberapa situasi, mungkin diperlukan untuk mengubah penerima fee ini, seperti ketika ada perubahan dalam manajemen pasar, penyesuaian kebijakan, atau pembaruan kontrak. **ModifyFeeRecipient** memberikan fleksibilitas untuk mengalihkan penerimaan fee ke akun lain yang lebih sesuai dengan kebutuhan saat ini.

Parameter Utama:

- **market**:
  - Pasar di mana penerima fee akan diubah. Ini mengidentifikasi pasar spesifik yang penerima fee-nya akan dimodifikasi.
- **newFeeRecipient**:
  - Alamat akun baru yang akan ditetapkan sebagai penerima fee. Ini adalah pengguna atau entitas yang akan mulai menerima fee yang dihasilkan dari aktivitas perdagangan di pasar tersebut.
- **currentFeeRecipient** (opsional):
  - Alamat akun penerima fee saat ini. Parameter ini dapat digunakan untuk memastikan bahwa perubahan penerima fee dilakukan secara sah oleh pihak yang berwenang.

Fungsi Utama:

Instruksi **ModifyFeeRecipient** berfungsi untuk mengubah akun yang akan menerima fee dari aktivitas perdagangan di pasar, memberikan fleksibilitas untuk mengelola pendapatan yang dihasilkan dari pasar. Berikut adalah beberapa fungsi utama dari instruksi ini:

1. **Pengalihan Penerima Fee yang Fleksibel**:
   - **ModifyFeeRecipient** memungkinkan pengelola pasar untuk mengubah penerima fee dengan mudah, menyesuaikan dengan perubahan dalam struktur manajemen, kebutuhan operasional, atau kebijakan pasar.
2. **Manajemen Pendapatan yang Lebih Baik**:
   - Dengan kemampuan untuk mengubah penerima fee, pasar dapat memastikan bahwa pendapatan dari fee dialokasikan ke akun yang paling sesuai dengan kebutuhan saat ini, seperti untuk mendanai pengembangan lebih lanjut atau untuk mendukung entitas baru yang mengambil alih manajemen pasar.
3. **Keamanan dan Kepastian**:
   - Instruksi ini memberikan kepastian bahwa hanya otoritas pasar yang berwenang yang dapat mengubah penerima fee, menjaga keamanan dan kepercayaan dalam pengelolaan pasar.
4. **Peningkatan Transparansi**:
   - Dengan dokumentasi dan pelacakan yang jelas dari perubahan penerima fee, instruksi ini membantu meningkatkan transparansi dalam pengelolaan keuangan pasar, sehingga semua pihak yang terlibat mengetahui siapa yang menerima pendapatan dari aktivitas perdagangan.
5. **Penyesuaian dengan Kebijakan Baru**:
   - Instruksi ini memungkinkan pasar untuk menyesuaikan penerima fee sesuai dengan kebijakan baru yang mungkin diterapkan oleh manajemen, seperti mengalihkan pendapatan ke badan amal,

mitra strategis, atau ke entitas yang baru saja bergabung dengan ekosistem pasar.

Contoh Penggunaan:

- Sebuah pasar di **Recto Verso** awalnya mengalihkan fee perdagangan ke akun pribadi pengelola pasar. Namun, setelah pasar diambil alih oleh entitas baru atau setelah restrukturisasi internal, pengelola pasar memutuskan untuk mengubah penerima fee menjadi akun perusahaan atau mitra strategis. Mereka menggunakan **ModifyFeeRecipient** untuk memperbarui penerima fee sesuai dengan perubahan ini.
- Dalam situasi lain, pengelola pasar memutuskan untuk menyumbangkan sebagian atau seluruh pendapatan dari fee ke organisasi non-profit sebagai bagian dari inisiatif sosial mereka. Dengan menggunakan **ModifyFeeRecipient**, mereka dapat mengarahkan pendapatan ke akun organisasi tersebut.