**Documentation for cronJobContronler.js**

**Overview**

This code file contains the implementation of the cronJobContronler function, which serves as "cron job" is a scheduled task, or command, that is executed automatically at a specific time or interval, using the cron utility on Unix-like operating systems, automating repetitive tasks like backups or system updates.and cron is a time-based job scheduler, meaning it allows you to schedule commands or scripts to run at specific times or intervals.

**Function: cron.schedule('\*/1 \* \* \* \*')**

cron.schedule('\*/1 \* \* \* \*', *async*() *=>* {

        await trxWingoController.addTrxWingo(1);

        await trxWingoController.handlingTrxWingo1P(1);

*const* [trxWingo] = await connection.execute(

          `SELECT \* FROM trx\_wingo\_game WHERE game = '${TRX\_WINGO\_GAME\_TYPE\_MAP.MIN\_1}' ORDER BY id DESC LIMIT 2`,

          [],

        );

*io*.emit("data-server-trx-wingo", { data: trxWingo });

        await winGoController.addWinGo(1);

        await winGoController.handlingWinGo1P(1);

*const* [winGo1] = await connection.execute('SELECT \* FROM `wingo` WHERE `game` = "wingo" ORDER BY `id` DESC LIMIT 2 ', []);

*const* data = winGo1; // Cầu mới chưa có kết quả

*io*.emit('data-server', { data: data });

*io*.emit('data-server-chat', { data: data, 'game': '1' });

        await k5Controller.add5D(1);

        await k5Controller.handling5D(1);

*const* [k5D] = await connection.execute('SELECT \* FROM d5 WHERE `game` = 1 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data2 = k5D; // Cầu mới chưa có kết quả

*io*.emit('data-server-5d', { data: data2, 'game': '1' });

*io*.emit('data-server-chat5d', { data: data2, 'game': '1' });

        await k3Controller.addK3(1);

        await k3Controller.handlingK3(1);

*const* [k3] = await connection.execute('SELECT \* FROM k3 WHERE `game` = 1 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data3 = k3; // Cầu mới chưa có kết quả

*io*.emit('data-server-k3', { data: data3, 'game': '1' });

*io*.emit('data-server-chatk3', { data: data3, 'game': '1' });

    });

This code runs a task every minute. Here’s what it does step by step:

1. It calls a function to add a transaction for a game.
2. It handles that transaction.
3. It retrieves the last two records from a database table for a game type and sends this data to the clients connected through WebSocket.
4. It repeats similar steps for another game, adding data, handling it, retrieving records, and sending them to clients.
5. It does this for four different games, each time sending the latest data to the connected clients.

In summary, the code keeps updating and sending the latest game data to users every minute.

**Function: cron.schedule('\*/3 \* \* \* \*')**

 cron.schedule('\*/3 \* \* \* \*', *async*() *=>* {

        await trxWingoController.addTrxWingo(3);

        await trxWingoController.handlingTrxWingo1P(3);

*const* [trxWingo] = await connection.execute(

          `SELECT \* FROM trx\_wingo\_game WHERE game = '${TRX\_WINGO\_GAME\_TYPE\_MAP.MIN\_3}' ORDER BY id DESC LIMIT 2`,

          [],

        );

*io*.emit("data-server-trx-wingo", { data: trxWingo });

        await winGoController.addWinGo(3);

        await winGoController.handlingWinGo1P(3);

*const* [winGo1] = await connection.execute('SELECT \* FROM `wingo` WHERE `game` = "wingo3" ORDER BY `id` DESC LIMIT 2 ', []);

*const* data = winGo1; // Cầu mới chưa có kết quả

*io*.emit('data-server', { data: data });

        await k5Controller.add5D(3);

        await k5Controller.handling5D(3);

*const* [k5D] = await connection.execute('SELECT \* FROM d5 WHERE `game` = 3 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data2 = k5D; // Cầu mới chưa có kết quả

*io*.emit('data-server-5d', { data: data2, 'game': '3' });

        await k3Controller.addK3(3);

        await k3Controller.handlingK3(3);

*const* [k3] = await connection.execute('SELECT \* FROM k3 WHERE `game` = 3 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data3 = k3; // Cầu mới chưa có kết quả

*io*.emit('data-server-k3', { data: data3, 'game': '3' });

    });

This code sets up a scheduled task that runs every 3 minutes. Inside this task, it does the following:

1. It calls functions to handle transactions for a game named "Trx Wingo" with the number 3.
2. It retrieves the last two records from a database table related to "Trx Wingo" and sends this data to clients connected through the io object.
3. It repeats similar steps for another game called "Win Go" with the number 3, retrieving the last two records from its table and sending this data to clients.
4. It does the same for a game called "K5" and another called "K3", fetching the last two records for each and sending the data to clients.

Overall, the code regularly updates connected clients with the latest game data every 3 minutes.

**Function: cron.schedule('\*/5 \* \* \* \*')**

 cron.schedule('\*/5 \* \* \* \*', *async*() *=>* {

        await trxWingoController.addTrxWingo(5);

        await trxWingoController.handlingTrxWingo1P(5);

*const* [trxWingo] = await connection.execute(

          `SELECT \* FROM trx\_wingo\_game WHERE game = '${TRX\_WINGO\_GAME\_TYPE\_MAP.MIN\_5}' ORDER BY id DESC LIMIT 2`,

          [],

        );

*io*.emit("data-server-trx-wingo", { data: trxWingo });

        await winGoController.addWinGo(5);

        await winGoController.handlingWinGo1P(5);

*const* [winGo1] = await connection.execute('SELECT \* FROM `wingo` WHERE `game` = "wingo5" ORDER BY `id` DESC LIMIT 2 ', []);

*const* data = winGo1; // Cầu mới chưa có kết quả

*io*.emit('data-server', { data: data });

        await k5Controller.add5D(5);

        await k5Controller.handling5D(5);

*const* [k5D] = await connection.execute('SELECT \* FROM d5 WHERE `game` = 5 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data2 = k5D; // Cầu mới chưa có kết quả

*io*.emit('data-server-5d', { data: data2, 'game': '5' });

        await k3Controller.addK3(5);

        await k3Controller.handlingK3(5);

*const* [k3] = await connection.execute('SELECT \* FROM k3 WHERE `game` = 5 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data3 = k3; // Cầu mới chưa có kết quả

*io*.emit('data-server-k3', { data: data3, 'game': '5' });

    });

This code sets up a scheduled task that runs every 5 minutes. In this task, it performs several actions:

1. It adds a transaction for a game called "Wingo" and handles it.
2. It retrieves the last two records from a database table for "Wingo" and sends this data to connected clients.
3. It adds and handles another game called "WinGo", retrieves its last two records from a database, and sends this data to clients.
4. It adds and handles a game called "5D", retrieves its last two records, and sends this data to clients.
5. Finally, it adds and handles a game called "K3", retrieves its last two records, and sends this data to clients.

Each time it runs, it checks the latest data for these games and shares it with users connected to the server.

**Function: cron.schedule('\*/10 \* \* \* \*')**

    cron.schedule('\*/10 \* \* \* \*', *async*() *=>* {

        await trxWingoController.addTrxWingo(10);

        await trxWingoController.handlingTrxWingo1P(10);

*const* [trxWingo] = await connection.execute(

          `SELECT \* FROM trx\_wingo\_game WHERE game = '${TRX\_WINGO\_GAME\_TYPE\_MAP.MIN\_10}' ORDER BY id DESC LIMIT 2`,

          [],

        );

*io*.emit("data-server-trx-wingo", { data: trxWingo });

        await winGoController.addWinGo(10);

        await winGoController.handlingWinGo1P(10);

*const* [winGo1] = await connection.execute('SELECT \* FROM `wingo` WHERE `game` = "wingo10" ORDER BY `id` DESC LIMIT 2 ', []);

*const* data = winGo1; // Cầu mới chưa có kết quả

*io*.emit('data-server', { data: data });

        await k5Controller.add5D(10);

        await k5Controller.handling5D(10);

*const* [k5D] = await connection.execute('SELECT \* FROM d5 WHERE `game` = 10 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data2 = k5D; // Cầu mới chưa có kết quả

*io*.emit('data-server-5d', { data: data2, 'game': '10' });

        await k3Controller.addK3(10);

        await k3Controller.handlingK3(10);

*const* [k3] = await connection.execute('SELECT \* FROM k3 WHERE `game` = 10 ORDER BY `id` DESC LIMIT 2 ', []);

*const* data3 = k3; // Cầu mới chưa có kết quả

*io*.emit('data-server-k3', { data: data3, 'game': '10' });

    });

This code sets up a scheduled task that runs every 10 minutes. Here's what it does step-by-step:

1. It calls a function to add a transaction for a game called "trxWingo" and processes it.
2. It retrieves the last two entries from the "trx*wingo*game" database for a specific game type and sends this data to the server.
3. It adds and processes another game called "WinGo" and retrieves the last two entries from the "wingo" database, then sends this data to the server.
4. It does something similar for a game called "5D", adding and processing it, then getting the last two entries from the "d5" database and sending that data to the server.
5. Finally, it adds and processes a game called "K3", retrieves the last two entries from the "k3" database, and sends that data to the server.

Each time it runs, the code updates and shares the latest game data with the server for different games.

**Function: cron.schedule('0 1 \* \* \*')**

cron.schedule('0 1 \* \* \*', *async*() *=>* {

        await connection.execute('UPDATE users SET roses\_today = ?', [0]);

        await connection.execute('UPDATE point\_list SET money = ?', [0]);

        await connection.execute('UPDATE turn\_over SET daily\_turn\_over = ?', [0]);

    });

This code sets up a scheduled task that runs every day at 1:00 AM. When it runs, it performs three actions in a database:

1. It resets the "roses\_today" value for all users to 0.
2. It resets the "money" value in the "point\_list" table to 0.
3. It resets the "daily*turn*over" value in the "turn\_over" table to 0.

These actions likely reset daily counters or values to start fresh for the new day.

**Function: cron.schedule("0 2 1 \* \*")**

cron.schedule("0 2 1 \* \*", *async* () *=>* {

      vipController.releaseVIPLevel();

    });

This code sets up a scheduled task using the cron library. It runs every month on the first day at 2:00 AM. When the task runs, it calls the releaseVIPLevel function from the vipController object. This likely means it updates or resets something related to VIP levels in a system.

**Function: cron.schedule("0 0 \* \* \*")**

cron.schedule(

      "0 0 \* \* \*",

*async* () *=>* {

        await winGoController.distributeCommission(),

        await k3Controller.distributeCommission(),

        await k5Controller.distributeCommission(),

        await trxWingoController.distributeCommission(),

        await vipController.releaseRebateCommission()

      }

    );

This code sets up a scheduled task that runs every day at midnight (00:00). When the task runs, it will perform several actions in order:

1. It will call the distributeCommission method from the winGoController.
2. Then, it will call the distributeCommission method from the k3Controller.
3. Next, it will call the distributeCommission method from the k5Controller.
4. After that, it will call the distributeCommission method from the trxWingoController.
5. Finally, it will call the releaseRebateCommission method from the vipController.

Each of these actions is done one after the other, and they are all marked as asynchronous, meaning the program will wait for each action to finish before moving to the next one