Bretton Woods *digital* Gold

Whitepaper DE/EN
Version 26.09.2023

Inhaltsverzeichnis

Einführung

Der Bretton Woods *digital* Gold Token (BWG) kombiniert die Blockchain-Technologie mit dem Edelmetall Gold. Der fungible Token ist zu 100% durch Gold gedeckt, das in einem Schweizer Zollfreilager gelagert wird.
Diese Bindung an das Gold bzw. den Goldpreis macht das BWG Token preisbeständig.

Gold für die Preisstabilität des Tokens

Historisch gesehen hat sich Gold als finanzielle Absicherung in unsicheren Zeiten bewährt. Die Preisstabilität des Edelmetalls lässt sich hervorragend als Inflationsschutz und zur Portfoliodiversifizierung nutzen.

Zukunftssichere Technologie

Der Token basiert auf dem Ethereum-Token-Standard ERC-777, der eine Weiterentwicklung des beliebten ERC-20-Standards ist und im Gegensatz zum ERC-20-Token verbesserte Funktionalitäten und volle Abwärtskompatibilität bietet. Durch die Veröffentlichung auf der Binance Smart Chain kann der BWG Token im Vergleich zu den meisten anderen Token kosten- und energieeffizienter gehandelt werden.

Gründe für den Bretton Woods *digital* Gold Token BWG

BWG Token , als tokenisiertes Gold, bietet die Möglichkeit, beliebig kleine Investitionen zu tätigen. Der Handel funktioniert in Echtzeit und kann weltweit zu jeder Tageszeit durchgeführt werden. Zugleich ist der Token ein Schritt in eine nachhaltige Zukunft. Nicht nur zur Verwahrung von Vermögenswerten, sondern auch zur Absicherung gegen instabile Währungen ist das BWG Token eine hervorragende Möglichkeit, seine Investitionen und Vermögenswerte abzusichern.

Mit dem BWG einen Schritt in die Zukunft wagen

Wir unterstützen Sie bei der sicheren Verwahrung des physischen Goldes und sorgen dafür, dass Sie jederzeit digital auf Ihr Gold zugreifen können. Durch die Abbildung auf der Binance Smart Chain ist es energieeffizienter und ressourcenschonender als herkömmliche Systeme und andere Blockchain-Netze - machen Sie mit uns den ersten Schritt in eine nachhaltige Zukunft.

Projektidee

Seit Jahrtausenden hat Gold für die Menschen eine besondere Anziehungskraft und Bedeutung. Neben der Verwendung als Schmuck oder Luxusgut dient es auch als Wertanlage.

Gerade Krisen haben gezeigt, dass Anleger in unsicheren Zeiten verstärkt Gold kaufen. Entscheidend ist dabei das Vertrauen in das Edelmetall in Krisenzeiten, denn Gold hat sich in der Vergangenheit als finanzielle Absicherung gegen den Verlust von Vermögenswerten bewährt, aber auch als Absicherung gegen Inflation aufgrund der weltweit stark gestiegenen Energiepreise. [1, 2]

Dabei stellen sich die Anleger die folgenden Fragen:

1. Wo kann ich schnell und einfach Gold kaufen?

2. Ist die Anlage in Gold abgesichert?

3. Wo wird das Gold gelagert?

Die Antwort ist der Bretton Woods *digital* Gold Token. Dieser kombiniert die Preisstabilität von Gold mit den technischen Vorteilen der Blockchain.

Die Idee des BWG

Durch die Kombination des klassischen physischen Assets Gold in Verbindung mit der Blockchain-Technologie ergeben sich verschiedene Vorteile. Die beschriebenen Vorteile von Gold bleiben auch in tokenisierter Form erhalten, jedoch kann z.B. die eingeschränkte Fungibilität aufgehoben werden. Das Ergebnis ist eine einfache und effiziente Möglichkeit, in Gold zu investieren. Die Preisstabilität des Tokens ergibt sich aus der Hinterlegung mit physischem Gold. Bei der Ausgabe des BWG-Tokens entspricht ein Token einem Gramm Gold. Um in den Genuss von Gold zu kommen, wird lediglich eine Krypto-Wallet (wie MetaMask) benötigt. Der manchmal komplexe Prozess, der mit dem Erwerb von physischem Gold verbunden ist, wird stark vereinfacht. Der Käufer des BWG wird Eigentümer des im Zollfreilager deponierten Goldes und muss sich nicht um dessen Verwahrung oder Transport an einen sicheren Ort kümmern. Die Lagerung in einem Schweizer Zollfreilager inklusive Zertifizierung ist im BWG enthalten.

Ein weiterer Vorteil ist, dass Gold aufgrund seines digitalen Charakters in beliebig[3] kleine Mengen aufgeteilt und in Echtzeit, grenzüberschreitend und zu jeder Tageszeit gehandelt werden kann. Bei derartigen Investitionen und Transaktionen stoßen die Anleger beim Kauf von physischem Gold in der Regel auf Probleme.

[1] Aye, G.C. et al. (2017): Does gold act as a hedge against inflaon in the UK? Evidence from a fraconal cointegraon approach over 1257 to 2016, Resources Policy, 54, pp. 53-57, doi: 10.1016/j.resourpol.2017.09.001.
[2] Federal Reserve Bank St. Louis (n.d.): Global price of Energy index (PNRGINDEXM). hps://fred.stlouisfed.org/series/ PNRGINDEXM, Zugriff am 23.02.2022.
[3] Beliebig bedeutet in diesem Fall 18 Dezimalstellen. Dies ist der Standardwert, der für das ERC-20-Token verwendet wird. Weitere Informationen unter: hps:// docs.openzeppelin.com/contracts/3.x/ERC-20.

Die Preisstabilität von Gold kann durch das BWG Token in instabilen Ländern und Entwicklungsländern als attraktives Instrument zur Absicherung gegen Hyperinflation und Instabilität genutzt werden. Das BWG Token bietet sich auch als Alternative zu anderen Stablecoins an. Laut Coinmarketcap sind diese in fast allen Fällen an den US-Dollar gekoppelt[4] . Damit stellt sich die Frage, was im Falle eines Wertverlustes des US-Dollars passiert. Um diesem potenziellen Problem entgegenzuwirken, bietet BWG eine Alternative zur Bindung an den US-Dollar. BWG ist bei einem Wertverlust des US-Dollars nicht betroffen, da es an den Goldpreis gekoppelt ist.

BWG als Investitionsschutz

Die Entstehung von Währungssystemen hat seit dem 19. Jahrhundert mehrere Phasen durchlaufen. Dazu gehörten beispielsweise der Goldstandard, Bretton Woods und das Kingstoner Währungssystem, das noch heute existiert.

Traditionell sind Geldsysteme um einen festen "Anker" herum aufgebaut. Jedes Zahlungsinstrument im Geldsystem ist letztlich an einen festen Betrag dieses Ankers gebunden. Der Anker kann viele Formen annehmen, wie z. B. die Bindung an ein Edelmetall oder eine Fiat-Währung. Während des Goldstandards war dies zum Beispiel Gold. Das bedeutete, dass jede von einer Regierung ausgegebene Währungseinheit in eine Goldeinheit konvertierbar war. Im Rahmen von Bretton Woods hielt dieser Anker sogar das gesamte internationale Währungssystem zusammen.
Alle teilnehmenden Länder stimmten festen Wechselkursen zum US-Dollar zu, und die Federal Reserve verpflichtete im Gegenzug die Zentralbanken aller teilnehmenden Länder, Dollar zu einem festen Kurs von 35 Dollar pro Feinunze in Gold zu tauschen. Derzeit ist der Anker in den meisten Währungssystemen eine von einer Regierung ausgegebene Fiat-Währung. Wachsende Ungleichgewichte in der Weltwirtschaft, Handelskriege, die Volatilität der Finanzmärkte, politische und soziale Spannungen in vielen Regionen der Welt sowie Kriege führen zu einer ständigen Instabilität der Währungen und Märkte. Das deutlichste Beispiel ist die Korona-Pandemie. Seit 2021 steigen die Preise stetig an. Aufgrund der strengen Beschränkungen und der Versorgungsengpässe konnte die Wirtschaft die Nachfrage in letzter Zeit nicht mehr decken. So erlitt der S&P 500 zu Beginn der Pandemie eine starke negative Korrektur .[5]

All diese Probleme führten in vielen Regionen der Welt zu Instabilität und großer Unsicherheit bei den Investoren.

Der Anstieg der Goldnachfrage während der Corona-Pandemie verstärkt die Wahrnehmung, dass Gold für Anleger weiterhin ein Zufluchtsort und Ankerwert ist.[6]

[4] Coinmarketcap (n.d.): Top stablecoin tokens by market capitalization, hps://coinmarketcap.com/de/view/stablecoin/, Abgerufen am 23. Februar 2022.
[5] Der S&P 500 ist ein Aktienindex. Visual Capitalist (2020): Veränderung der Performance des S&P 500 während der COVID-19-Pandemie im Vergleich zu früheren großen Crashs, Stand: August 2020, https://www.statista.com/statistics/1175227/s-and-p-500-major-crashes-change/, abgerufen am 23. Februar 2022.
[6] World Gold Council (n.d.): Statistiken zu Goldangebot und -nachfrage, https://www.gold.org/goldhub/data/gold-supply-and-demand- statistics, abgerufen am 23. Februar 2022.

Dienstleistungen

Verwahrung Ihres Goldes

Wir garantieren die Sicherheit des hinterlegten Goldes. Jede Wertmarke wird mit zertifiziertem Gold in einem Zollfreilager in der Schweiz hinterlegt.

Unkomplizierte Verarbeitung

Mit einem einfachen Knopfdruck können Sie Gold[1] kaufen oder verkaufen, ohne sich Gedanken über Lieferung, Lagerung, Absicherung und Bearbeitungsgebühren zu machen.

Zugang zu Ihrem Vermögen

Sie haben jederzeit Zugriff auf Ihre BWG-Token, da die Token in Ihrer eigenen Wallet gespeichert sind. Darüber hinaus können Sie auf unserer Plattform jederzeit verschiedene Informationen zu Ihrer angeschlossenen Wallet und Ihrem Konto einsehen.

Prüfung des Goldes durch regelmäßige Audits

Wir werden regelmäßig Audits durchführen, um die Quantität und Qualität des gelagerten Goldes, mit dem die Token unterlegt sind, zu überprüfen. Wir werden die Prüfberichte auf unseren Websites veröffentlichen.

Rückkauf von BWG-Token

Die Token können vom Kunden auf der Plattform gegen FIAT-Währung zurückverkauft werden. Der Verkaufspreis basiert auf dem aktuellen LME (London Metal Exchange) zuzüglich einer Bearbeitungsgebühr.

[1] Der Verkauf Ihrer BWG-Token auf der BWG-Plattform wird bald verfügbar sein.

Ausgabe und Verkauf von Token

Um zu gewährleisten, dass nur durch Gold gedeckte Token im Umlauf sind, werden nur so viele Token über die Plattform verkauft, wie physisches Gold im Schweizer Zollfreilager gelagert ist. Bis neues Gold erworben, gelagert und zertifiziert ist, kann es daher in Zeiten hoher Nachfrage zu kurzen Verzögerungen beim Verkauf kommen.

Beim Kauf von BWG-Token über die Plattform werden die Token unmittelbar nach Zahlungseingang an die angeschlossene, nicht depotgebundene Wallet des Kunden gesendet. Somit stellt der Kauf von BWG eine Übertragung des Eigentums an dem hinterlegten Gold dar.

Um den gesetzlichen Anforderungen, z. B. im Bereich der Geldwäsche, gerecht zu werden, müssen die Kunden zunächst den KYC-Prozess (Know Your Customer) durchlaufen. Die Kunden müssen gültige Ausweisdokumente hochladen und der KYC-Anbieter prüft, ob der Kunde zum Kauf von Token berechtigt ist.

Die für das Gold anfallenden Lagerkosten werden automatisch per Smart Contract aus der Wallet des Kunden beglichen. Die Lagerkosten belaufen sich auf ca. 0,22% der gehaltenen Token pro Monat.
Die Kosten können je nach den Vertragsbedingungen mit dem Zolllager leicht variieren. Die Plattform erhebt über einen Smart Contract automatisch Gebühren in Höhe von 0,66% pro Quartal von der Wallet des Kunden, um die technische Funktionalität sicherzustellen.

Technische Details

Smart Contract

Smart Contracts sind einfach Programme, die auf einer Blockchain gespeichert sind und ausgeführt werden, wenn vorher festgelegte Bedingungen erfüllt sind. Smart Contracts für Token sind nicht nur für die Erstellung von Token verantwortlich, sondern auch für die Abwicklung von Transaktionen und die Verfolgung des Guthabens der einzelnen Token-Inhaber. Um Token zu erhalten, muss man einige ETH/BNB an den Token Contract senden und erhält im Gegenzug Token zugeteilt.

ERC-20 Token-Standard

ERC-20 ist der technische Standard für fungible Token, die mit der EVM-Blockchain erstellt werden. Der Standard ermöglicht die Implementierung einer Standard-API für Token innerhalb von Smart Contracts. Dieser Standard bietet grundlegende Funktionen für die Übertragung von Token sowie für die Freigabe von Token mit niedrigem Wert, damit sie von einer anderen dritten Partei auf der Blockchain ausgegeben werden können.

ERC-777 Token-Standard

Dieser Standard definiert eine neue Art der Interaktion mit einem Token Contract und ist gleichzeitig rückwärtskompatibel mit ERC-20. Er definiert erweiterte Funktionen für die Interaktion mit Token. Er ermöglicht es Betreibern, Token im Namen einer anderen Adresse, eines Contracts oder eines regulären Kontos zu versenden.

BWG-Token-Contract

Bretton Woods Digital Gold (BWG) ist ein ERC-777-Standard-Token. Dieser Token wird in der EVM-basierten Binance Smart Chain eingesetzt. Der BWG-Token ist durch physisches Gold gedeckt. Dieses Gold haben wir gekauft und sicher gelagert.

Im Gegensatz zu ähnlichen Projekten basiert unsere Deckung nicht nur auf Optionen zum Kauf von Gold. Stattdessen sichert tatsächlich vorhandenes physisches Gold die Token.

Alle Tokeninhaber müssen vierteljährlich Lagerkosten für das Gold bezahlen. Diese Kosten beziehen sich auf das Gold, das die von ihnen gehaltenen Token absichert.

Warum wir uns für den ERC-777-Standard entschieden haben

BWG ist ein goldgedeckter Token, bei dem 1 BWG-Token 1 Gramm Gold entspricht, so dass der Token-Inhaber für die Lagerungskosten aufkommen muss. Der ERC-777-Standard verfügt über eine Standardoperatorfunktionalität; der Standardoperator kann Token im Namen einer anderen Adresse versenden. Wir verwenden den ERC-777-Standardoperator, um die Lagerkosten vierteljährlich von den Konten der Token-Inhaber abzuziehen; der Smart Contract Anpassungen des Abzugszeitraums zulässt.

Token Implementierung

Wir haben die folgenden drei grundlegenden Smart Contracts implementiert, um diesen ERC-777 -Token und die Funktionalität für den Lagerkostenabzugsprozess zu erstellen.

1. BWGToken Contract
2. StorageOperator Contract
3. TokenTransferOperator Contract

Wir haben einen weiteren Smart Contract implementiert, der darauf abzielt, automatisch eine erforderliche Mindestmenge an Token auf der Hotwallet der Bretton Woods Gold Plattform zu halten, indem wir Chainlink-Keeper verwenden. Damit soll die Sicherheit der Hotwallet verbessert werden, indem die Menge der dort befindlichen Mittel zu einem bestimmten Zeitpunkt begrenzt wird.

4. HotwalletTransferer Contract

BWGToken Contract

Der BWG-Token ist mit dem ERC-777-Standard implementiert und kann in EVM-basierten Blockchains eingesetzt werden. Die grundlegenden Merkmale des Vertrags sind:

1. Übertragung von Token von einem Konto auf ein anderes. (ERC-20 Standard)
2. Ermittelt den aktuellen Token-Saldo eines Kontos. (ERC-20-Standard)
3. Ermittelt den Gesamtbestand der im Netzwerk verfügbaren Token. (ERC-20-Standard)
4. Genehmigung, ob eine Menge von Token von einem Konto durch ein Drittkonto ausgegeben werden kann. (ERC-20-Standard)
5. Übertragen von Token auf die Empfänger-Wallet, ohne eine Bestätigung vom Token-Inhaber zu erhalten, indem Sie den Standard-Operator verwenden. (ERC-777-Standard)
6. Wir verwenden den Standardoperator als Smart Contract (anstelle einer regulären Wallet) namens TokenTransferOperator und überschreiben die Funktion revokeOperator, damit der Token-Inhaber die Standardoperatoren nicht widerrufen kann. Dies ermöglicht es uns, den Mechanismus zum Abzug der Lagerkosten zu sichern. (Benutzerdefinierte Funktion)
7. Alle Token-Inhaber werden als storageWallet-Liste im _beforeTokenTransfer-Hook gespeichert, um die Lagerkosten abzuziehen. (Benutzerdefinierte Funktion)
8. Verwalten einer costfreeWallet-Liste, die keine Abzüge für Lagerkosten zulässt. Die kostenfreien Wallets werden nur für interne Wallets der BrettonWoods *digital* AG verwendet, um den Abzug von Lagerkosten von internen Wallets zu verhindern und somit unnötige Transaktionsgebühren zu vermeiden. (Benutzerdefinierte Funktion)
9. Autorisieren des StorageOperator-Contracts. (Benutzerdefinierte Funktion)
10. Wir überschreiben _callTokensToSend und _callTokensReceived mit einem leeren Body, so dass die Sende- und Empfangshooks nicht ausgeführt werden. (Benutzerdefinierte Implementierung)
11. Die Rollen admin und maintainer sind berechtigt, diesen Vertrag zu aktualisieren. (Zugangskontrolle)
12. Überschreiben von revokeRole und renounceRole, um Admin-Benutzer einzuschränken. (Zugriffskontrolle)

StorageOperator Contract

Der StorageOperator-Contract ist ein Art Verwaltungsvertrag, der für die Abrechnung der Lagerkosten der Token auf Basis monatlicher / vierteljährlicher / halbjährlicher Abrechnungen zuständig ist. Die grundlegenden Merkmale des StorageOperator Contracts sind die folgenden.

1. Der Contract speichert die Zustände fundAddress, batchSize, storageCostRate, transactionFee, performDay, timeDuration, minTokenBalance und lastPerformTimestamp.

2. Der Contract initiiert batchProcess an einem bestimmten Datum des Monats mit einem bestimmten Monatsintervall.

3. Führt batchSend durch, indem eine Liste von Wallets und berechneten Kosten mit einer anderen Methode namens getCostableWalletBatch erhält. Die batchSend-Methode überträgt Token, um die Lagerkosten zu bezahlen, indem sie die operatorSend-Funktion des TokenTransfer-Operator Contracts verwendet. Dieser Vorgang wird so lange wiederholt, bis das Ende der costableWallets erreicht ist.

4. Wir können den StorageOperator Contract neu bereitstellen, wenn wir die Verwaltungsfunktionen aktualisieren müssen, und dann BWGToken und TokenTransferOperator autorisieren.

5. Die Rollen admin, maintainer und batch-executor sind berechtigt, diesen Contract zu aktualisieren. (Zugriffskontrolle)

6. Überschreiben von revokeRole und renounceRole, um Admin-Benutzer einzuschränken. (Zugriffskontrolle)

TokenTransferOperator Contract

Der TokenTransferOperator ist der Standardoperator des BWGToken Contracts und die operatorSend-Methode dieses Contracts kann Token im Namen des Token-Inhabers versenden.
Die grundlegenden Merkmale des TokenTransferOperator Contracts sind:

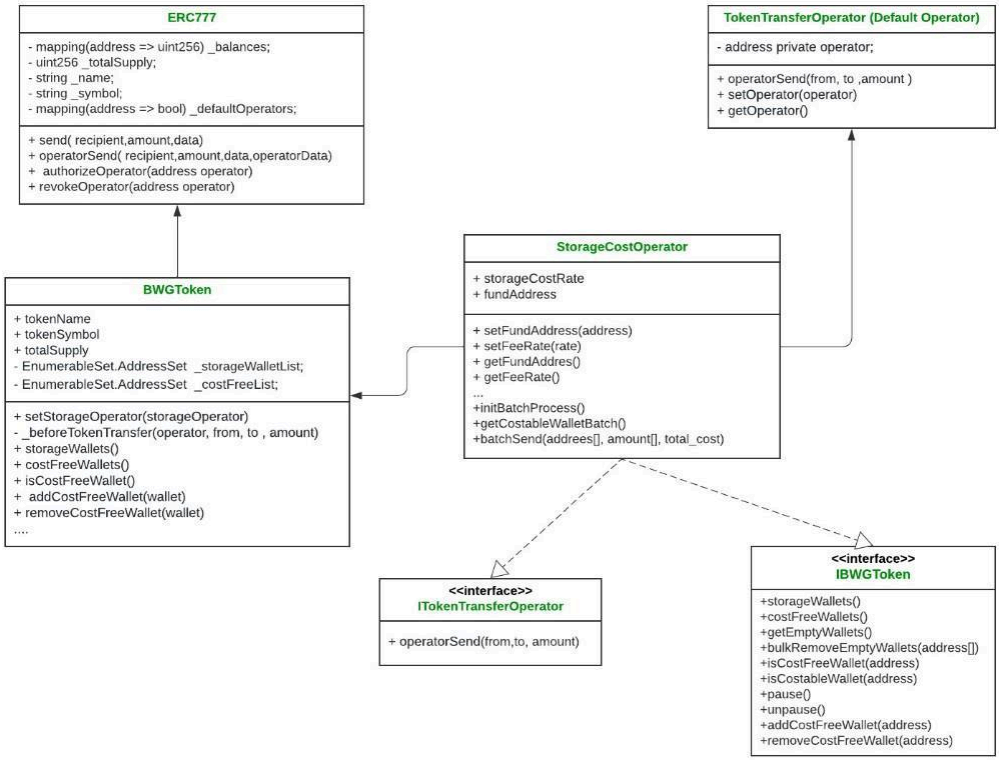
1. StorageOperator Contract autorisieren.

2. Führt operatorSend aus, um Token im Namen des Token-Inhabers zu übertragen. Nur StorageOperator Contacts dürfen auf die operatorSend-Methode zugreifen.

3. Die Rollen admin und maintainer sind berechtigt, diesen Contract zu aktualisieren. (Zugangskontrolle)

4. Überschreiben von revokeRole und renounceRole, um Admin-Benutzer einzuschränken. (Zugriffskontrolle)

BWG Token Contract UML diagram



HotwalletTransferer Contract

Der HotwalletTransferer Contract wurde entwickelt, um automatisch die erforderliche Mindestmenge an Token auf der Hotwallet der Bretton Woods Gold-Plattform aufrechtzuerhalten, indem Chainlink-Keeper verwendet werden. Chainlink-Keeper benötigen einige LINK-Token, um diesen Smart Contract zu betreiben. Daher sorgt der Systemadministrator für ein ausreichendes Guthaben an LINK-Token auf dem Konto des Keepers. Jedes Chainlink-Konto hat ein Mindestguthaben an LINK-Token, um sicherzustellen, dass das Konto auch bei einem plötzlichen Anstieg des Guthabens weiterläuft. Wenn Ihr LINK-Guthaben unter diesen Betrag fällt, wird der Upkeep nicht durchgeführt. Die Grundfunktionen des HotwalletTransferer Contracts sind:

1. Automatischer Transfer von Token von einer Herkunftswallet (Token-Halter / Cold-Wallet) zu einer anderen Wallet (Hot-Wallet), wenn der Kontostand der Ziel-Wallet unter einem bestimmten Schwellenwert liegt.

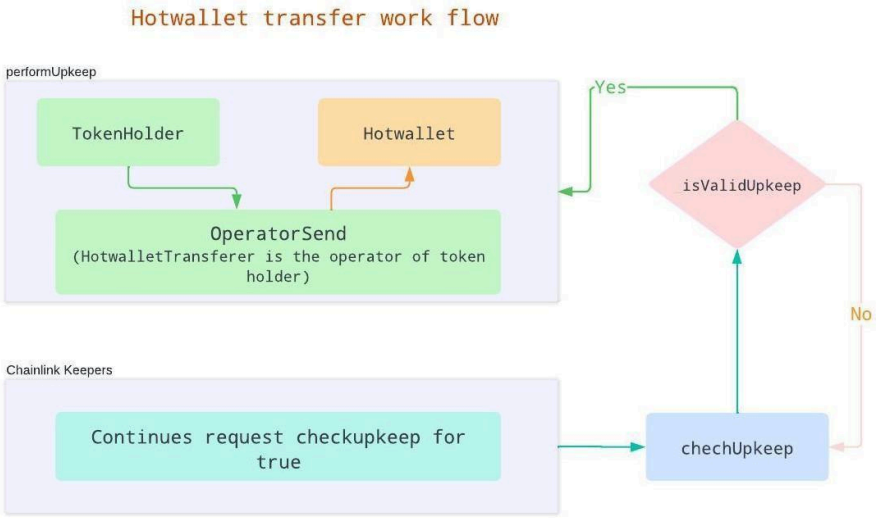
2. Die Chainlink-Keeper sind dafür verantwortlich, diesen Contract zu automatisieren.

3. Die Herkunfts- und Zielwallet können geändert werden.

4. Die Halter von Chainlink Keepern müssen über genügend LINK-Token verfügen, um übertragen zu können.

5. Die Rollen admin und maintainer sind berechtigt, diesen Contract zu aktualisieren. (Zugangskontrolle)

6. Überschreiben von revokeRole und renounceRole, um Admin-Benutzer einzuschränken. (Zugriffskontrolle)



HotWalletTransfer Empfehlung

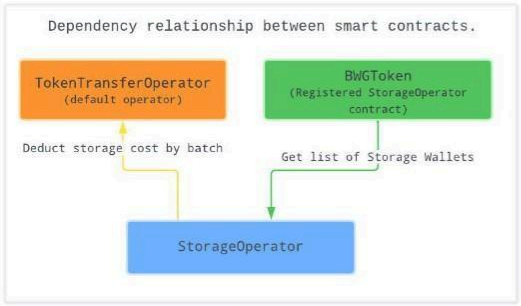
1. **Wechsel des Token-Inhabers:** Vor dem Wechsel des Token-Inhabers sollte der Systemadministrator den neuen Inhaber für den HotWalletTransferer Contract autorisieren, um einen fehlerfreien Transfer zu gewährleisten. Der performUpkeep schlägt weiterhin fehl und das System verliert LINK-Token, wenn es nicht vor dem Wechsel des Tokeninhabers einen neuen Inhaber autorisiert.

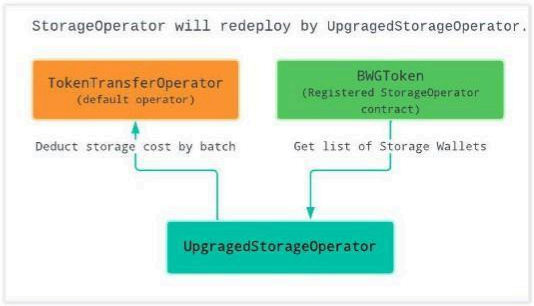
2. **Sicherstellung eines Mindestguthabens an LINK-Token:** Chainlink-Keeper benötigen einige LINK-Token, um diesen Smart Contract zu betreiben, so dass wir sicherstellen einen positiven Saldo aufzuweisen.

StorageOperator Umstrukturierung

Wir haben die Möglichkeit, den StorageOperator Contract neu einzusetzen, wenn wir Verwaltungsfunktionen aktualisieren, z. B. die Einstellungen für den Abzug von Lagerkosten, die von der vorhandenen Implementierung nicht abgedeckt werden kann. Der Beispielcontract"UpgradedStorageOperator" und das Bereitstellungsskript befinden sich im Ordner "redeploy_scrips".

Abhängigkeitsverhältnis zwischen Smart Contracts.

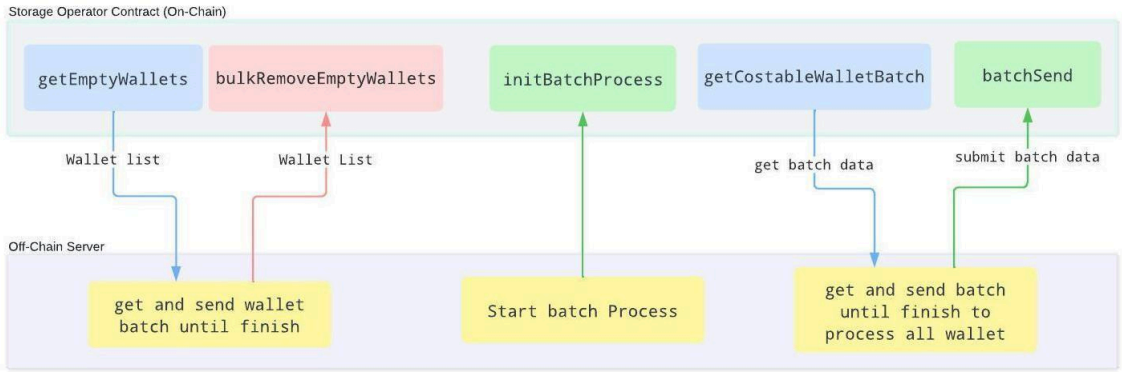




Abzug von Goldlagerkosten (Batch-Verfahren)

Der StorageOperator Contract hat die Fähigkeit, Lagerkosten einzuziehen, indem er den TokenTransferOperator Contract mit der batchSend-Methode verwendet. Zunächst starten Off-Chain-Server den Batch-Prozess, indem sie eine initBatchProcess-Anforderung anfordern. Dann wird die batchSend-Methode ausgelöst, indem Wallets und Beträge von der getCostableWallet-Batch-Methode gesammelt werden. Dieser Prozess wird bis zum letzten Index der storageWallet-Liste fortgesetzt.
Die folgenden Vorgänge müssen während des Batch-Prozesses durchgeführt werden:

1. Abzug der Lagerkosten von allen Token-Inhabern, wenn diese nicht in der costfreeWallet sind.
2. Unterbrechung der Tokenübertragung nach Beginn der initBatchProcess-Anforderung und Aufhebung der Unterbrechung nach deren Beendigung.
3. Der batch-executor kann den Batch-Prozess ausführen (initBatchProcess und batchSend).
4. Der batch-executor kann die Liste der leeren Wallets aus den storageWallets entfernen.
5. Der maintainer kann während des Batch-Prozesses keinen StorageOperator Contract Zustand aktualisieren.
6. Sammeln des Restsaldos nach dem Abzug, wenn er zu gering ist oder unter dem Mindestschwellenwert für den Tokensaldo liegt.



Kommunikation zwischen Off-Chain-Server und StorageOperator während eines Batch-Prozesses.

Wenn ein Batch Process durchgeführt wird

Standardmäßig wird der Batch-Prozess alle drei Monate (vierteljährlich) mit einem festen Monat wie (Januar, April, Juli, Oktober) ausgeführt. Er hängt nicht vom Einsatzdatum des Contracts ab. Obwohl timeDuration konfigurierbar ist, kann es monatlich, vierteljährlich, halbjährlich und jährlich eingestellt werden. Wenn timeDuration auf monatlich eingestellt ist, wird der Batch-Prozess jeden Monat zum angegebenen Datum ausgeführt.

Mechanismus zum Schutz vor Dust (kleine Menge an BWG-Token)

Für den Abzug der Lagergebühren haben wir einen statischen Wert namens minTokenBalance im Smart Contract. Dieser Wert wird auf einen vernünftigen Schwellenwert gesetzt, um zu verhindern, dass kleine Mengen von BWG-Token in einer Adresse liegen und Lagergebühren generieren, während ihr Wert zu gering ist, um für irgendetwas effektiv genutzt zu werden.

Das bedeutet, dass wir während des Abzugs von Lagergebühren von einem Nutzer alle verfügbaren Token von dieser Wallet abziehen, wenn der verbleibende Betrag nach einem Abzug geringer ist als der Mindestschwellenwert für das Token-Guthaben.

Leere Wallets aus der Walletsliste entfernen

StorageOperator hat die Möglichkeit, mit Hilfe des BWGToken Contract eine leere Wallet aus der Liste der Lagerwallets zu entfernen. Dies kann vor oder nach dem Batch-Prozess geschehen. Zunächst sammelt der On-Chain-Server eine Liste des ersten verfügbaren Batch leerer Wallets aus der getEmptyWallets-Methode und sendet sie dann an bulkRemoveEmptyWallets.

Umgang mit Costfree Wallets

Die costfree (kostenfreie) Wallet ist eine spezielle Art von Wallet, bei der die darin enthaltenen Token nicht für die Lagerkostenabzüge berücksichtigt werden. Der maintainer kann costfree Wallets hinzufügen oder entfernen. Die costfree Wallets werden nur für interne Wallets der BrettonWoods *digital* AG verwendet, um Lagerkostenabzüge von internen Wallets und damit unnötige Transaktionsgebühren zu vermeiden. Die Bretton Woods *digital* AG muss die Lagerkosten ohnehin direkt mit den Lagerpartnern abrechnen und braucht dafür keine Token aus einer Firmen-Wallet zu sammeln.

Rollenbasierter Zugriff

Die Zugriffskontrolle bietet einen allgemeinen rollenbasierten Zugriffskontrollmechanismus. Es können mehrere hierarchische Rollen erstellt und zugewiesen werden, jede für mehrere Konten. Jede Rolle kann nur die Aktionen (Ausführungsfunktionen) ausführen, die ihr zugewiesen wurden. Eine Rolle kann mehreren Wallets zugewiesen werden. Ein Administrator kann andere Rollen gewähren oder entziehen.

Für den BWGToken Contract haben wir zum Beispiel zwei verschiedene Rollen eingeführt:

Eine heißt admin (für zusätzliche Sicherheit auf gnosis safe multisig wallet eingestellt) und die andere heißt maintainer. Der Administrator kann jeder Wallet jede Rolle zuweisen oder entziehen. Admin ist eine gnosis safe Wallet, die die ihr zugewiesenen Aktionen ausführen kann, wobei sie die Zustimmung mehrerer Besitzer benötigt. Auf der Grundlage von Smart Contracts werden verschiedene Rollen erstellt und für verschiedene Aufgaben zugewiesen.

- Der BWGToken Contract hat die Rollen admin und maintainer.
- Der StorageOperator Contract hat die Rollen admin, maintainer und batch-executor.
- Der TokenTransferOperator Contract hat die Rollen admin und maintainer.
- Der HotWalletTransferer Contract hat die Rollen admin und maintainer.

Admin: Ein admin kann anderen Rollen, einschließlich der Rolle des admin, Rechte gewähren oder entziehen. Aber er kann seine eigene Rolle nicht widerrufen oder aufgeben. Im Allgemeinen kann ein admin eine Cold Wallet oder eine Gnosis Safe Multisig Wallet sein.

Maintainer: Der maintainer kann bestimmte Zustände des Smart Contracts hinzufügen/aktualisieren/entfernen. Das kann eine einzelne Wallet sein.

Batch-executor: Der batch-executor kann einen monatlichen Batch-Prozess durchführen. Es sollte eine individuelle Wallet sein, damit ein automatisiertes Skript den Batch-Prozess in einem bestimmten Zeitintervall abwickeln kann.

Die Liste der Operationen kann je nach Benutzerrolle durchgeführt werden.

Rolle	Typ	Rolle gewähren durch	Verzicht auf die Rolle	Rolle widerruf en durch	Smart Contract
admin	Gnosis Safe Multisig-Wallet	admin, maintainer und batch executor	– Anderer admin – Mainta iner und batch executor	Nein	BWGToken, StorageOperator, TokenTransferOperator HotWalletTransferer
maintainer	Omdovodieööe Waölet	Nein	Nein	Ja	BWGToken, StorageOperator, TokenTransferOperator HotWalletTransferer
batch-executor	Individuelle Wallet	Nein	Nein	Ja	StorageOperator

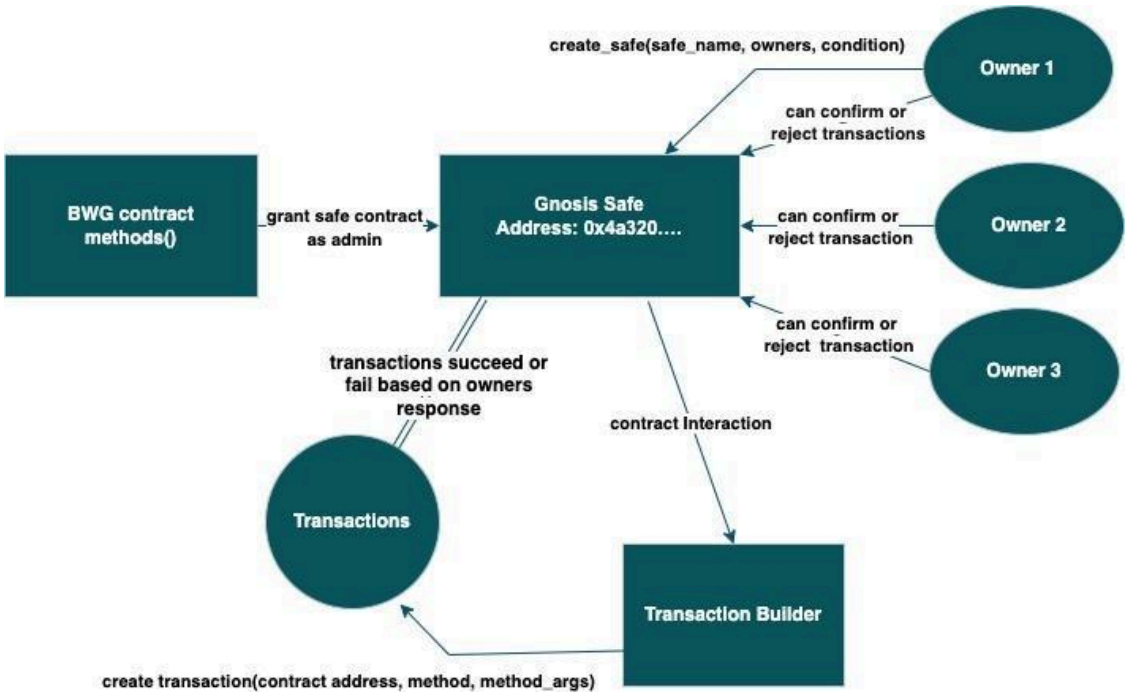
Gnosis Safe

Gnosis Safe ist eine Smart Contract Wallet, die auf Ethereum läuft und eine Mindestanzahl von Personen benötigt, um eine Transaktion zu genehmigen, bevor sie stattfinden kann.

Wie funktioniert das?

Gnosis Safe ist eine Multi-Signatur Smart Contract Wallet, die es Nutzern ermöglicht, eine Liste von Eigentümern/Unterzeichnern und eine Mindestanzahl von Unterzeichnern zu definieren, die zur Bestätigung einer Transaktion erforderlich sind. Sobald der Schwellenwert an Eigentümerkonten eine Transaktion bestätigt hat, kann die Safe-Transaktion ausgeführt werden.

Multisig (z.B. 2 von 3 Eigentümern) Transaktion



BWG-Token Contract Übersicht

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
allowance	Read/Lesen	Adressbesitzer, Adressabgeber	Nein	Gibt die verbleibende Anzahl von Token zurück, die der Ausgeber im Namen des Besitzers durch transferFrom ausgeben darf. Dieser Wert ist standardmäßig Null.
balanceOf	Read/Lesen	Adresskonto	Nein	Gibt die Anzahl der Token zurück, die sich im Besitz des Kontos befinden.
decimals	Read/Lesen		Nein	Gibt die Anzahl der Dezimalstellen zurück, die verwendet wurden, um die Benutzerdarstellung zu erhalten.
defaultOperators	Read/Lesen		Nein	Gibt die Liste der Standardoperatoren zurück. Diese Konten sind Operatoren für alle Token-Inhaber, auch wenn authorizeOperator nie für sie aufgerufen wurde.
granularity	Read/Lesen		Nein	Gibt den kleinsten Teil des Tokens zurück, der nicht teilbar ist. Das bedeutet, dass alle Token-Operationen (Erstellung, Bewegung und Zerstörung) Beträge haben müssen, die ein Vielfaches dieser Zahl sind
Name	Read/Lesen		Nein	Gibt den Namen des Tokens zurück.
symbol	Read/Lesen		Nein	Gibt das Symbol des Tokens zurück.
totalSupply	Read/Lesen		Nein	Gibt den Gesamtvorrat des Tokens zurück.
costfreeWallets	Read/Lesen		Nein (storageOperator)	Gibt die Liste der costfreeWallets des Tokens zurück.
storageWallets	Read/Lesen		Nein (storageOperator)	Gibt die Liste der storageWallets bei Aufruf von storageOperator zurück, ansonsten leere Liste.
isCostableWallet	Read/Lesen		Nein	Gibt true zurück, wenn der Token-Inhaber nicht in der costfree-Liste steht, sonst false.
isCostfreeWallet	Read/Lesen		Nein	Gibt true zurück, wenn die „Wallet in die costfree Liste aufgenommen wurde, sonst false.
getBatchSize	Read/Lesen		Nein	Die externe Methode wird verwendet, um die aktuelle Batch-Größe zu ermitteln.
getEmptyWallets	Read/Lesen		Nein (storageOperator)	Die Methode getEmptyWallets wird verwendet, um alle verfügbaren Wallets mit einem

				leeren Guthaben zu finden.
addCostfreeWallet	Write/Schreiben	address wallet	Nein (storageOperator)	Die Methode wird verwendet, um die Wallet zur Liste der kostenfreien Wallets hinzuzufügen und ein AddCostFreeWallet Event auszulösen. Sendet ein AddCostFreeWallet Event aus.
removeCostfreeWallet	Write/Schreiben	address wallet	Nein (storageOperator)	Die Methode wird verwendet, um die Wallet aus der Liste der kostenfreien Wallets zu entfernen und das Event RemoveCostFreeWallet auszulösen. Sendet ein Event "RemoveCostFreeWallet" aus.
bulkRemoveEmptyWallets	Write/Schreiben	address [] wallet	Nein (storageOperator)	Die Methode wird verwendet, um Wallets mit Nullsaldo aus der Liste storageWallet zu entfernen.

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
setStorageOperator	Write/Schreiben	Address storage_operator_address	maintainer	Die Methode wird verwendet, um die Adressen von Lageroperator Contracts zu aktualisieren, damit gültige Contacts Zugriff auf die Liste der Walletsn erhalten.
Gibt ein StorageOperator-Event aus."	Read/Lesen	Address account	Nein	Gibt die Anzahl der Token zurück, die sich im Besitz des Kontos befinden.
setBatchSize	Write/Schreiben	uint256	maintainer	Die externe Methode wird zur Aktualisierung der Batch-Größe verwendet.
Gibt ein BatchSize Event aus."	Read/Lesen		Nein	Gibt die Liste der Standardoperatoren zurück. Diese Konten sind Operatoren für alle Token-Inhaber, auch wenn authorizeOperator nie für sie aufgerufen wurde.
bulkSend	Write/Schreiben	address[] wallets, uint256[] costs	Nein	Die Methode wird verwendet, um Token in einer einzigen Transaktion an mehrere Wallets zu senden
pause	Write/Schreiben		Nein (storageOperator)	Die Methode wird verwendet, um die Übertragung von Token zu unterbrechen. Gibt ein Pause Event aus. Anforderungen: der Vertrag darf nicht pausiert werden.
unpause	Write/Schreiben		Nein (storageOperator)	Die Methode wird verwendet, um die Pause bei der Tokenübertragung aufzuheben Gibt ein Unpaused Event aus. Anforderungen: Der Contract muss pausiert werden.
burn	Write/Schreiben	uint256 amount, bytes data	Nein	Zerstört Token vom Konto des Callers, wodurch der Gesamtbestand verringert wird.

				Gibt ein Burned Event aus. Anforderungen: Der Caller muss mindestens eine bestimmte Anzahl von Token haben."

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
send.	Write/Schreiben	address recipient, uint256 amount, bytes data	Nein	Überträgt den Betrag der Token vom Konto des Callers an den Empfänger. Sendet ein Event. Anforderungen: Der Caller muss mindestens über eine bestimmte Anzahl von Token verfügen. Der Empfänger kann nicht die Null Adresse sein.
operatorSend	Write/Schreiben	address sender, address recipient, uint256 amount, bytes data, bytes operatorData	Nein	Verschiebt die Menge der Token vom Sender zum Empfänger. Der Aufrufer muss ein Operator des Absenders sein. Sendet ein Event aus. Anforderungen: Der Absender kann nicht die Null Adresse sein. Der Absender muss mindestens eine bestimmte Anzahl von Token haben. Der Anrufer muss ein Operator für den Absender sein.
authorizeOperator	Write/Schreiben	address operator	Nein	Macht ein Konto zu einem Operator des Callers.. Gibt ein AuthorizedOperator-Ereignis aus. Anforderungen: Operator kann nicht die Caller-Adresse sein.
transfer	Write/Schreiben	address recipient, uint256 amount	Nein	Verschiebt die Tokenmenge vom Konto des Callers zum Empfänger. Gibt einen booleschen Wert zurück, der angibt, ob der Vorgang erfolgreich war. Sendet ein Transfer Event.
approve	Write/Schreiben	address spender, uint256 amount	Nein	Legt amount als Freibetrag des Senders über die Token des Callers fest. Gibt einen booleschen Wert zurück, der angibt, ob der Vorgang erfolgreich war.

transferFrom	Write/Schreiben	address sender, address recipient, uint256 amount	Nein	Verschiebt den Betrag der Token vom Absender zum Empfänger unter Verwendung des Vergütungsmechanismus. Der Betrag wird dann von der Vergütung des Callers abgezogen. Gibt einen booleschen Wert zurück, der angibt, ob der Vorgang erfolgreich war. Sendet ein Übertragungs-Event
revokeRole	Write/Schreiben	bytes32 role, address account	admin	Durch die Überschreibung der Zugriffskontrolle revokeRole können sie alle Rollen außer der Adminrolle widerrufen.
renounceRole	Write/Schreiben	bytes32 role, address account	admin	Override Access control renounceRole ermöglicht es ihnen, auf alle Rollen außer der Adminrolle zu verzichten.

BWG Token Events

Benutzerdefinierte Ereignisse:

StorageOperator(address wallet)

AddCostFreeWallet(address wallet)

RemoveCostFreeWallet(address wallet)

RemoveCostableWallet(address wallet)

BatchSize(uint256 batchSize)

ERC-777 Events:
Sent(address operator, address from, address to, uint256 amount, bytes data, bytes operatorData)
Minted(address operator, address to, uint256 amount, bytes data, bytes operatorData)
Burned(address operator, address from, uint256 amount, bytes data, bytes operatorData)
AuthorizedOperator(address operator, address tokenHolder)
RevokedOperator(address operator, address tokenHolder)

ERC-20 Events:

Transfer(address from, address to, uint256 value)

Approval(address owner, address spender, uint256 value)

Pausable

Paused(address account)

Unpaused(address account)

StorageOperator Contract Übersicht

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
storageWalletCount	Lesen		Nein	Gibt die Anzahl der Wallets von Token-Inhaber zurück.
costFreeWallets	Lesen		Nein	Gibt die Liste der Wallets der Token-Inhaber zurück, die keine Lagerkosten zahlen.
costFreeWalletCount	Lesen		Nein	Gibt die Anzahl der Wallets der Token-Inhaber zurück, die keine Lagerkosten zahlen.
getBatchCursor	Lesen		Nein	Methode, die verwendet wird, um den lastIndex des aktuellen Batch-Cursors während des Batch-Prozesses zu ermitteln.
getBatchSize	Lesen		Nein	Methode, die zur Ermittlung der Chargengröße des Batch-Prozesses verwendet wird.
getCostableWalletBat ch	Lesen		Nein	Externe Methode, die verwendet wird, um alle Informationen über den nächsten Batch zu erhalten. Gibt zurück: address[] wallets, uint256[] costs, uint256 totalCost, bool isCompleted **Valid batch request required.
getFundAddress	Lesen		Nein	Gibt aktualisierte funcaddress zurück.
getLastPerformTimes tamp	Lesen		Nein	Rückgabe des zuletzt aktualisierten Zeitstempels der Batch-Prozessausführung.
getMinTokenBalance	Lesen		Nein	Rückgabe des zuletzt aktualisierten Wertes von minTokenBalance .
getPerformDay	Lesen		Nein	Gibt Tag des Monas des Batch-Prozesess zurück.
getPerformTimeDurat ion	Lesen		Nein	Gibt Interval des Batch-Prozesses zurück. Wie 1: Monatlich 3: Vierteljährlich 6: Halbjährlich 12: Jährlich
getStorageCostRate	Lesen		Nein	Rückgabe der StorageCostRate des Batch-Prozesses.
getTransactionFee	Lesen		Nein	Gibt die durchschnittliche Überweisungsgebühr jedes Users während des Batch-Prozesses abgezogen.

getEmptyWallets	Lesen		Nein	Externe Methode, die verwendet wird, um das erste verfügbare leere Wallet-Array zu erhalten

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
getStorageOverview	Lesen		Nein	Gibt alle alle Informationen über den anstehenden Batch-Prozess zurück. uint256 totalStorageCost, uint256 totalWallet, uint256 totalWalletUnderThreshold, uint256 totalCostUnderThreshold, uint256 date, uint256 duration, uint256 lastPerformTime, uint256 costRate
pause	Lesen		Maintainer	Die Methode wird verwendet, um Token-Transfers zu pausieren Sendet ein Pausenevent. Anforderungen: • der Contract darf nicht pausiert sein.
unpause	Schreiben		Maintainer	Die Methode wird verwendet, um pausierte Token-Transfers nicht mehr zu pausieren. Gibt ein Unpaused-Event aus. Anforderungen: • der Contract muss pausiert sein.
bulkRemoveEmptyWallets	Schreiben	address[] calldata wallets	BatchExecutor	Externe Methode, die zum Entfernen aus der Storage Wallet verwendet wird. **Valid batch request required
initBatchProcess	Schreiben	uint256 walletMaxlimit	BatchExecutor	Externe Methode, mit der der Batch-Prozess eingeleitet wird. Sendet ein InitBatchProcess-Event **Valid batch request required
batchSend	Schreiben	address[] calldata wallets, uint256[] calldata costs, uint256 currentIndex, uint256 totalCost	BatchExecutor	Externe Methode zur Durchführung von Batch-Prozessen zur Erhebung von Lagergebühren Gibt ein StorageCostSummary-Event aus. **Valid batch request required

setStorageCostRate	Schreiben	uint256 storageCostRate	Maintainer	Externe Methode wird verwendet, um storageCostRate einzustellen. Die Fließkommazahl darf nicht als Rate festgelegt werden, so dass die Rate immer das 100-fache der ursprünglichen Rate beträgt. Gibt ein StorageCostRate-Event aus.
setTransactionFee	Schreiben	uint256 transactionFee	Maintainer	Die externe Methode wird zur Aktualisierung von transactionFee verwendet. Sendet ein TransactionFee-Event.

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
setFundAddress	Schreiben	address fundAddress	Maintainer	Diese externe Methode wird zur Aktualisierung der fundAddress verwendet, an die die monatlichen **Lagerkosten** überwiesen werden. Gibt ein FundAddress-Event aus.
setLastPerform-Timestamp	Schreiben	uint256 timestamp	Maintainer	Die externe Methode wird zur Aktualisierung von lastPerformTimestamp verwendet. • Zum Ändern von lastPerformTimestamp ist die die Rolle maintainer erforderlich. • newTimestamp sollte nicht größer sein als der Zeitstempel des Blocks. • Darf während des Batch-Prozesses nicht ausgeführt werden. Sendet ein LastPerformTimestamp-Event.
setMinTokenBalance	Schreiben	uint256 minTokenBalance	Maintainer	Die externe Methode wird zur Aktualisierung von minTokenBalance verwendet. • Maintainer Rolle erforderlich, um minTokenBalance zu ändern. • Darf während des Batch-Prozesses nicht ausgeführt werden. Gibt ein MinTokenBalance-Event aus.
setPerformTime-Duration	Schreiben	uint256 duration	Maintainer	Die externe Methode wird verwendet, um das Intervall des Batch-Prozesses zu aktualisieren. Sendet ein PerformTimeDuration-Event.
setPerformDay	Schreiben	uint256 dayOfMonth	Maintainer	Die externe Methode wird zur Aktualisierung von performDay (Tag des Monats) verwendet. Sendet ein PerformDay-Event aus.
setBatchSize	Schreiben	uint256 batchSize	Maintainer	Die externe Methode wird verwendet, um die Transaktionsgröße batchSize zu setzen. Gibt ein BatchSize-Event aus.
addCostFreeWallet	Schreiben	address wallet	Maintainer	Die Methode addCostFreeWallet wird verwendet, um eine kostenfreie Wallet zur Liste hinzuzufügen. • Maintainer Rolle erforderlich, um minTokenBalance zu ändern. • Darf während des Batch-Prozesses nicht ausgeführt werden.
removeCost-FreeWallet	Schreiben	address wallet	Maintainer	Die Methode removeCostFreeWallet wird verwendet, um eine Wallet aus der Liste der kostenpflichtigen Wallets zu entfernen. • Zum Ändern von minTokenBalance ist die Erlaubnis des Maintainers erforderlich. • Darf während des Batch-Prozesses nicht ausgeführt werden.

revokeRole	Schreiben	bytes32 role, address account	Admin	Überschreiben der Zugriffskontrolle revokeRole ermöglicht es ihnen, alle Rollen außer der Adminrolle zu widerrufen
renounceRole	Schreiben	address fundAddress	Admin	Überschreiben der Zugriffkonstrolle renounceRole ermöglicht es Ihnen, auf alle Rollen außer der Admin-Rolle zu verzichten.

StorageOperator Events

InitBatchProcess(uint256 maxWallet)
LastPerformTimestamp(uint256 timestamp)
MinTokenBalance(uint256 minBalance)
PerformTimeDuration(uint256 duration)
PerformDay(uint256 day)
StorageCostRate(uint256 storageCostRate)
FundAddress(address fundAddress)
BatchSize(uint256 batchSize)
TransactionFee(uint256 fee)
StorageCostSummary(address fundAddress, uint256 storageCostRate, uint256 totalFee)

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
getOperator	Lesen		Nein	Gibt die Adresse des Lagerhaltungskosten Operator Contracts zurück.
setOperator	Lesen	address operator	Ja	Die externe Methode wird zur Einstellung des Operators verwendet. Gibt ein Operator-Event aus.
operatorSend	Lesen	IERC-777 token, address from, address to, uint256 amount	Ja	Der OperatorSend wird verwendet, um Token im Namen von Token-Inhabern zu übertragen.
revokeRole	Schreiben	bytes32 role, address account	Admin	Durch die Überschreibung der Zugriffskontrolle revokeRole können sie alle Rollen außer der Adminrolle widerrufen.
renounceRole	Schreiben	bytes32 role, address account	Admin	Override Access control renounceRole ermöglicht es ihnen, auf alle Rollen außer der Adminrolle zu verzichten.

TokenTransferOperator Events

Operator(address operator)

HotWalletTransferer Contract Überblick

Name der Methode	Anfrage Typ	Parameter	Rolle	Kommentare
checkUpkeep	Lesen		Nein	Die checkUpkeep Methode ist eine externe KeeperCompatibleInterface Methode, mit der geprüft wird, ob upkeepNeeded erforderlich ist.
performUpkeep	Schreiben		Nein	Die performUpkeep Methode ist eine externe KeeperCompatibleInterface Methode, die verwendet wird, um performUpkeep auszuführen, wenn sie von der checkUpkeep Methode true erhält. Gibt ein TransferToHotWallet-Event aus.
setHolderAddress	Schreiben	address newHolder	Maintainer	Die externe Methode wird verwendet, um den Halter des Tokens zu aktualisieren. Gibt ein HolderAddress-Event aus.
setHotWalletAddress	Schreiben	address hotwallet	Maintainer	Die externe Methode wird zur Aktualisierung der Hotwallet-Adresse verwendet. Gibt ein HotWalletAddress-Event aus.
revokeRole	Schreiben	bytes32 role, address account	Admin	Durch die Überschreibung der Zugriffskontrolle revokeRole können alle Rollen außer der Adminrolle widerrufen.
renounceRole	Schreiben	bytes32 role, address account	Admin	Override Access control renounceRole erlaubt es, auf alle Rollen außer der Adminrolle zu verzichten.

HotWalletTransferer Events

TransferToHotWallet(address indexed holder, address indexed hotWallet, uint256 value)

HotWalletAddress(address indexed hotWalletAddress)

HolderAddress(address indexed holder)

TriggerTokensAmount(uint256 triggerTokensAmount)

TokensAmountToAdd(uint256 tokensAmountToAdd)

Coverage Bericht

Datei	Stmts%	Branch	Func%	Zeile%	Uncovered Lines
Contracts	**100**	**97.41**	**100**	**100**	
BWGToken.sol	100	95.83	100	100	
HotWalletTransferer.sol	100	100	100	100	
StorageOperator.sol	100	97.73	100	100	
TokenTransferOperator.sol	100	100	100	100	

** Bitte verwenden Sie die Nodeversion v16.13.0, um Coverage Berichte zu erhalten. Es funktioniert nicht mit der neuesten Node-Version.

Table of Content

Introduction

The Bretton Woods *digital* Gold Token (BWG) combines blockchain technology with the precious metal gold. The fungible token is 100% backed by gold, which is stored in a Swiss bonded warehouse.
This link to gold or the gold price makes the BWG pricestable.

Gold for price stability of the token

Historically, gold has proven its worth as a financial hedge in uncertain times. Accordingly, the price stability of the precious metal can be used excellently as an inflation hedge and for portfolio diversification.

Future-proof technology

The token is based on the Ethereum token standard ERC-777, which is an evolution of the popular ERC-20 standard and, unlike the ERC-20 token, offers improved functiona- lities and full backward compatibility. By publishing it on the Binance Smart Chain, the BWG can be traded in a more cost and energy efficient way compared to most
Gold Backed Tokens.

Reasons for the Bretton Woods *digital* Gold Token BWG

BWG, as tokenized gold, offers the possibility to make investments as small as you like. Trading works in real time and can be carried out worldwide at any time of day. At the same time, the token is a step into a sustainable future. Not only as a safekeeping of assets, but also for hedging against unstable currencies, BWG is an excellent way to hedge one's investments and assets.

Taking a step into the future with BWG

We support you in the safekeeping of the physical gold and ensure that you can access your gold digitally at any time. By mapping it on the Binance Smart Chain it is more energy efficient and resource saving than traditional systems and other blockchain networks.Take the first step into a sustainable future with us.

Project idea

For thousands of years, gold has had a special attraction and significance for people. In addition to its use as jewelry or luxury goods, it also serves as an investment.

Especially crises have shown that investors start to buy more gold in uncertain times. The decisive factor here is confidence in the precious metal during the acute phase of the crisis, as gold has proven itself in the past as a financial hedge against the loss of assets, and also as a hedge against inflation due to the sharp rise in energy prices worldwide. [1, 2]

In doing so, investors ask themselves the following questions:

1. Where can I buy gold quickly and easily?
2. Is the investment in gold hedged?
3. Where is the gold stored?

The answer is the Bretton Woods Digital Gold Token. This combines the price stability of gold with the technical advantages of the blockchain.

Idea of BWG

Various advantages result from the combination of the classic physical asset gold in combination with blockchain technology. The described advantages of gold also remain in tokenized form, but its limited fungibility, for example, can be eliminated. The result is a simple and efficient way to invest in gold. The price stability of the token results from the deposit with physical gold. When the BWG Token is issued, one token is equivalent to one gram of gold. In order to benefit from gold, only a crypto wallet (like MetaMask) is required. The sometimes complex process associated with the di- rect acquisition of physical gold is greatly simplified. The buyer of the BWG becomes the owner of the deposited gold in the bonded warehouse and does not have to take care of their safekeeping or transport to a safe place. Storage in a Swiss bonded warehouse including certification is included with the BWG.

Another advantage is that due to the digital character of gold, it can be divided into arbitrary[3] small quantities and traded in real time, across borders and at any time of day. For investments and transactions like this, investors usually face problems when buying physicial gold.

[1] Aye, G.C. et al. (2017): Does gold act as a hedge against inflaon in the UK? Evidence from a fraconal cointegraon approach over 1257 to 2016, Resources Policy, 54, pp. 53-57, doi: 10.1016/j.resourpol.2017.09.001.
[2] Federal Reserve Bank St. Louis (n.d.): Global price of Energy index (PNRGINDEXM). hps://fred.stlouisfed.org/series/ PNRGINDEXM, accessed 02/23/2022.
[3] Arbitrary in this case means 18 decimal places. This is the default value used for the ERC-20 token. More information at: hps://docs.openzeppelin.com/contracts/3.x/ERC-20.

The price stability of gold can be used by the BWG in unstable and developing countries as an attractive vehicle to hedge against hyperinflation and instability. The BWG also offers itself as an alternative to other stablecoins. According to Coinmarketcap, these are linked to the US dollar in almost all cases[4] . This poses the die risicated question of what happens in the event of a decline in the value of the U.S. dollar. To counteract this potential problem, the BWG offers an alternative to the US dollar peg. The BWG is not affected in the event of a decline in the value of the U.S. dollar because it is pegged to the gold price.

BWG as Investment protection

The emergence of monetary systems has gone through several phases since the 19th century. These included, for example, the gold standard, Bretton Woods, and the Kingston monetary system, which still exists today.

Traditionally, monetary systems are built around a fixed "anchor." Each payment instrument in the monetary system is ultimately tied to a fixed amount of this anchor. The anchor can take many forms, such as being tied to a precious metal or a fiat currency. During the gold standard, for example, this was gold. This meant that every unit of currency issued by a go- vernment was convertible into a unit of gold. In fact, under Bretton Woods, this anchor held the entire international monetary system together.
All participating countries agreed to fixed exchange rates to the U.S. dollar, and the Federal Reserve, in return, committed central banks of all participating countries to exchange dollars for gold at a fixed rate of $35 per troy ounce. Currently, the anchor in most monetary systems is a fiat currency issued by a government. Growing imbalances in the global economy, trade wars, volatility in financial markets, political and social tensions in many regions of the world, as well as wars, are continuously causing instability in currencies and markets. The most re- cent example has been the corona pandemic. Since 2021 prices have been rising constantly. Due to the severe restrictions as well as supply bottlenecks, the economy recently failed to meet demand. Thus, the S&P 500 suffered from a strong negative correction at the beginning of the pandemic[5] .

All these problems led to instability and great uncertainty among investors in many regions of the world.

An increase in demand for gold during the corona pandemic reinforces the perception that gold remains a haven and anchor asset for investors.[6]

[4] Coinmarketcap (n.d.): Top stablecoin tokens by market capitalization, hps://coinmarketcap.com/en/view/stablecoin/, Retrieved Feb. 23, 2022.
[5] Visual Capitalist (2020): Change in performance of S&P 500 during COVID-19 pandemic vs previous major crashes as of August 2020, https://www.statista.com/statistics/1175227/s-and-p-500-major-crashes-change/, Retrieved Feb. 23, 2022.
[6] World Gold Council (n.d.): Gold supply and demand statistics, https://www.gold.org/goldhub/data/gold-supply-and-demand- statistics, Retrieved Feb. 23, 2022.

Services

Safekeeping of your gold

We guarantee the security of the deposited gold. Each token is deposited with certified gold in a bonded warehouse in Switzerland.

Uncomplicated processing

With a simple push of a button, you can buy or sell gold[1] without worrying about its delivery, storage, hedging, and handling fees.

Access to your assets

You have access to your BWG Tokens at any time, as the tokens are stored in your own wallet. In addition, various information about your connected wallet and account is available for you to view at any time on our platform.

Audit of the gold by regular Audits of the gold

We will have regular audits to establish the quantity and quality of the stored gold that is backing the tokens. We will publish the audit reports on our websites.

Buyback of BWG tokens

The tokens can be sold back on the platform by the customer against FIAT currency. The selling price is based on the current LME (London Metal Exchange) price plus a handling fee.

[1] Selling your BWG Tokens on the BWG Platform will be available soon.

Token issue and sale

To guarantee that only tokens backed by gold are in circulation, only as many tokens are sold via the platform as there is physical gold stored in the Swiss bonded warehou- se. Therefore, until new gold is acquired, stored and certified, there may be short delays in sales during periods of high demand.

When BWG tokens are purchased via the platform, the tokens are sent to the customer's connected non-custodial wallet immediately after receipt of payment. Thus, the purchase of BWG constitutes a transfer of ownership of the deposited gold.

In order to comply with legal requirements, such as in the area of money laundering, customers must first complete the Know Your Customer (KYC) process. Customers have to upload valid identification documents and the KYC provider checks whether the customer is eligible to purchase tokens.

The storage costs incurred for the gold are automatically settled via smart contract from the customer's wallet. The storage costs amount to approximately 0,22% of the tokens held each month.
The cost may vary slightly depending on the terms of the contract with the bonded warehouse. The platform automatically charges 0,66% of the customer's wallet each quarter via smart contract to ensure technical functionality.

Technical Details

Smart Contract

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. Token smart contracts are not only responsible for creating tokens but also for handling transactions and keeping track of the balances of each token holder. To get some tokens one has to send some ETH/BNB to the token's contract in return for allocated tokens.

ERC-20 Token Standard

ERC-20 is the technical standard for fungible tokens created using the EVM blockchain. The standard allows for the implementation of a standard API for tokens within smart contracts. This standard provides basic functionality to transfer tokens, as well as al- low tokens to be approved so they can be spent by another on-chain third party.

ERC-777 Token Standard

This standard defines a new way to interact with a token contract while remaining backward compatible with ERC-20. It defines advanced features to interact with tokens. Namely, it allows operators to send tokens on behalf of another address, contract or regular account.

BWG Token Contract

Bretton Woods Digital Gold (BWG) is an ERC-777 standard gold backed token deployed in the EVM based Binance Smart Chain. Since the token is secured by physical gold we have purchased and stored, instead of just options to buy gold like similar projects, all token holders need to pay storage costs for the gold backing the tokens they hold, on a quarterly basis.

Why We Chose The ERC-777 Standard

BWG is a gold backed token where 1 BWG token is 1 gram of gold so that the token holder has to pay for storage cost. ERC-777 standard has a default operator functiona- lity; the default operator can send tokens on behalf of another address. We use the ERC- 777 default operator to deduct storage cost from token holder accounts on a quarterly basis, though the smart contract does allow for adjustments to the deduction period.

Token Implementation

We implemented the following three basic smart contracts to create this ERC-777 ba- sed token and the functionality for the storage cost deduction process.

1. BWGToken Contract
2. StorageOperator Contract
3. TokenTransferOperator Contract

We have implemented another contract that is designed to automatically maintain a required minimum amount of tokens on the hotwallet of the Bretton Woods Gold platform, using Chainlink keepers. This is to improve the security of the hotwallet by limi- ting the amount of funds there at any given point.

4. HotwalletTransferer Contract

BWGToken Contract

The BWG token is implemented with the ERC-777 standard and can be deployed in EVM based blockchains. The basic features of the contract are:

1. Transfer tokens from one account to another. (ERC-20 standard)
2. Get the current token balance of an account. (ERC-20 standard)
3. Get the total supply of tokens available on the network. (ERC-20 standard)
4. Approve whether an amount of tokens from an account can be spent by a third-party account. (ERC-20 standard)
5. Transfer tokens to the recipient wallet without getting confirmation from the token holder using the default operator. (ERC-777 standard)
6. We use the default operator as a smart contract (instead of a regular wallet) called **TokenTransferOperator** and we override the function **revokeOperator**, so that the token holder cannot revoke default operators. This allows us to secure the storage cost deduction mechanism. (Custom function)
7. Store all token holders as a **storageWallet** list in **_beforeTokenTransfer** hook to deduct storage cost. (Custom function)
8. Manage **costfreeWallet** list, that will not allow deductions for storage costs. The cost free wallets will only be used for internal wallets of the BrettonWoods *digital* AG, to prevent storage cost deductions from internal wallets and thus prevent unnecessary transaction fees. (Custom function)
9. Authorize **StorageOperator** contract. (Custom function)
10. We override **_callTokensToSend** and **_callTokensReceived** with an empty body, so that it does not execute the sending and receiving hooks.
    (Custom implementation)
11. **Admin** and **maintainer** roles are allowed to update this contract. (Access Control)
12. Override **revokeRole** and **renounceRole** to restrict admin users. (Access Control)

StorageOperator Contract

The **StorageOperator** contract is an administrative contract responsible for the deduc- tion of token storage costs on the basis of monthly / quarterly / half yearly transacti- ons. The Basic features of the **StorageOperator** contract are the following.

1. Contract stores **fundAddress, batchSize, storageCostRate, transactionFee, performDay, timeDuration, minTokenBalance** and **lastPerformTimestamp** states.

2. Contract initiates **batchProcess** on a specific date of the month with a specific monthly interval.

3. Perform **batchSend** by getting a list of wallets and calculated costs using another method called **getCostableWalletBatch**. The **batchSend** method transfers tokens to pay for storage costs, by using the **operatorSend** function of the **TokenTransfer-Operator** contract. This process will repeat until the end of the **costableWallets** is reached.

4. We can redeploy the **StorageOperator** contract in case of needing to update administrative functionality and then authorize **BWGToken** and **TokenTransferOperator**.

5. **Admin, maintainer** and **batch-executor** roles are allowed to update this contract. (Access Control)

6. Override **revokeRole** and **renounceRole** to restrict **admin** users. (Access Control)

TokenTransferOperator Contract

The **TokenTransferOperator** is the default operator of the **BWGToken** contract and the **operatorSend** method of this contract can send tokens on behalf of the token holder. The basic features of the token transfer operator contract are:

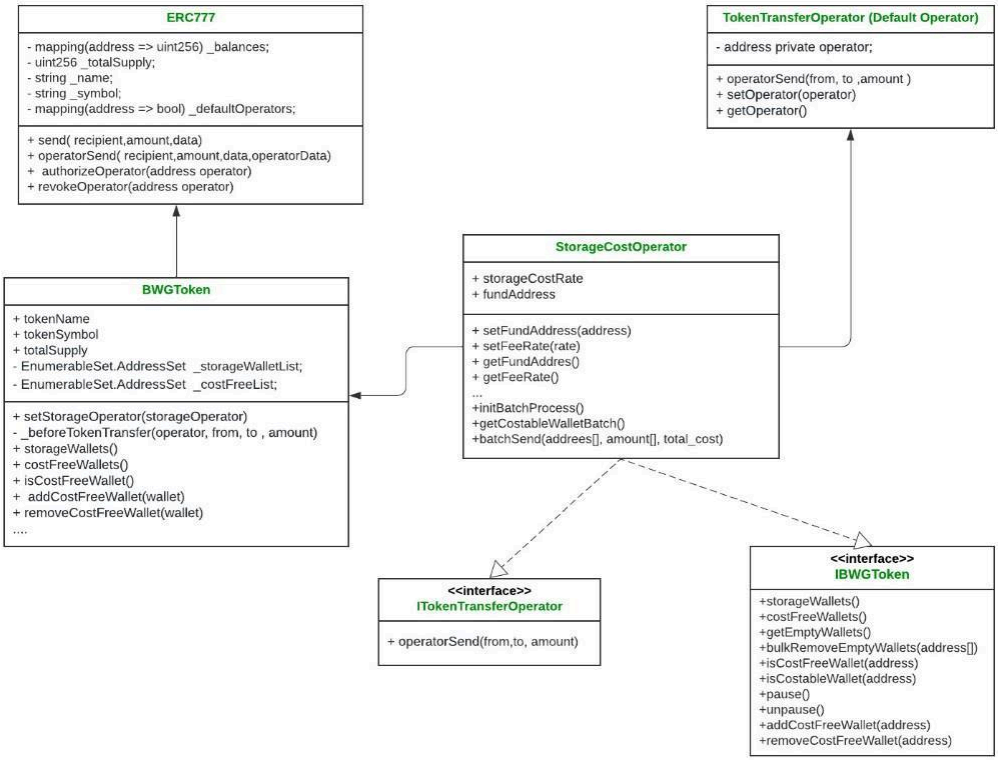
1. Authorize **StorageOperator** contract.

2. Perform **operatorSend** to transfer token on behalf of token holder. Only **Storage-Operator** contracts are allowed to access the **operatorSend** method.

3. **Admin** and **maintainer** roles are allowed to update this contract. (Access Control)

4. Override **revokeRole** and **renounceRole** to restrict **admin** users. (Access Control)

BWG Token Contract UML diagram



HotwalletTransferer Contract

The **HotwalletTransferer** contract is designed to automatically maintain the required minimum amount of tokens on the balance of the hotwallet of the Bretton Woods Gold platform, by using Chainlink keepers. Chainlink keepers require some LINK tokens to operate this smart contract, so the system admin ensures a sufficient balance of LINK tokens in the keepers account. Each Chainlink upkeep has a minimum balance to ensu- re that an upkeep will still run, should a sudden spike occur. If your upkeep LINK balance drops below this amount, the upkeep will not be performed. The Basic features of the **HotwalletTransferer** contract are:

1. Automatically transfer tokens from a source wallet (token holder / cold wallet) to another wallet (hotwallet) if the destination wallet balance is under a set threshold.

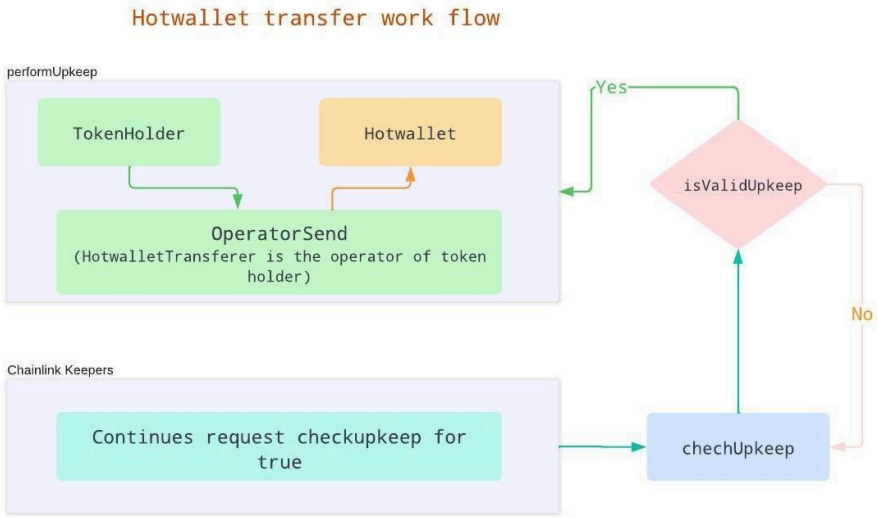
2. Chainlink keepers are responsible to automate this contract.

3. The source wallet and destination wallet can be changed.

4. Chainlink keepers need to have sufficient LINK tokens in order to transfer.

5. **Admin** and **maintainer** roles are allowed to update this contract. (Access Control)

6. Override **revokeRole** and **renounceRole** to restrict **admin** users. (Access Control)



HotWalletTransfer Recommendation

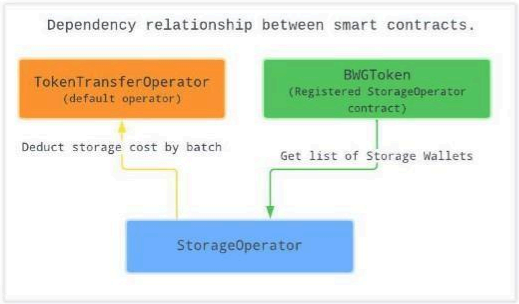
1. **Changing token holder**: Before changing token holders, the system **admin** should authorize new holders to the **HotWalletTransferer** contract for error free transfer service. The **performUpkeep** continues to fail and the system loses LINK tokens if it does not authorize a new holder before changing token holder.

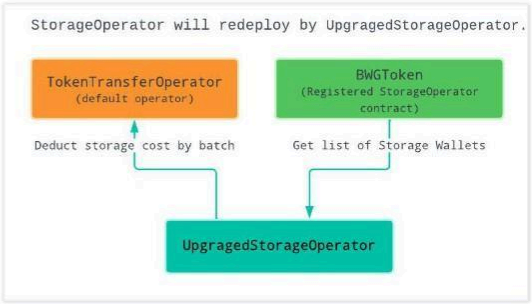
2. **Ensure minimum LINK token balance**: Chainlink keepers require some LINK tokens to operate this smart contract, so that we will assure that the upkeep will maintain a positive balance

StorageOperator Redeployment

We have the ability to redeploy the **StorageOperator** contract in case of updating administrative functionality like storage cost deduction policy that is not able to be covered by the existing implementation. The sample **UpgradedStorageOperator** contract and deployment script remain in the **redeploy_scrips** folder.

Dependency relationship between smart contracts.





Gold Storage Cost Deduction (batch process)

The **StorageOperator** contract has the ability to collect storage costs by using the **TokenTransferOperator** contract with the **batchSend** method. First of all, off-chain ser- vers start the batch process by requesting an **initBatchProcess** request. Then triggering the **batchSend** method by collecting wallets and amounts from the **getCostableWallet- Batch** method. This process continues until the final index of the **storageWallet** list.
The following operation need to be done during the batch process:

1. Deduct storage costs from all token holders if holders are not in the **costfreeWallet** list.

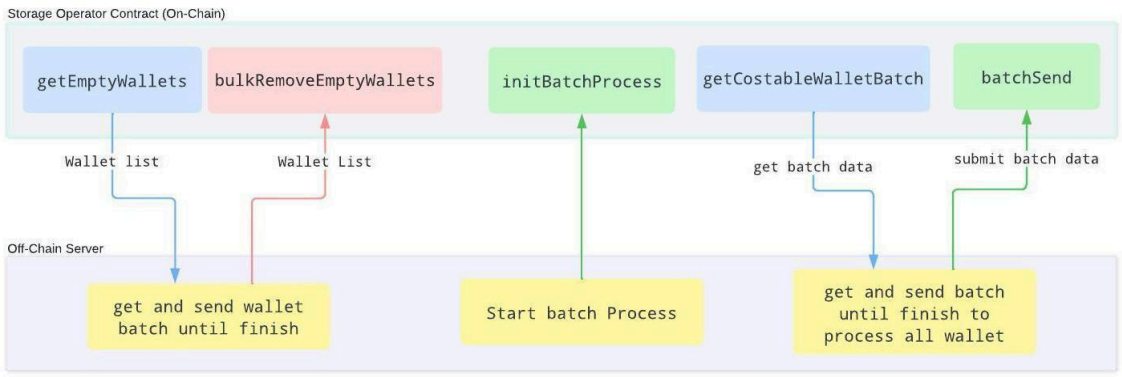
2. Pause token transfer after the **initBatchProcess** request starts and unpause after it finishes.

3. The batch-executor can execute the batch process (**initBatchProcess** and **batch-Send**).

4. The **batch-executor** can remove the list of **emptyWallets** from the **storageWallets** list.

5. The **maintainer** can't update any **StorageOperator** contract state during the batch process.

6. Collect the remaining balance after deduction if it is too little or less than the mini-mum threshold token balance.



Communication between off-chain server and **StorageOperator** during a batch process.

When Performing a Batch Process

By default the batch process will execute every three month interval (quarterly) with a fixed month like (January, April, July, October). It does not depend on the deployment date of the contract. Although **timeDuration** is configurable it could be set monthly, quarterly, half yearly and yearly. If **timeDuration** is set to monthly, the batch process will run at the specified date each month.

Dust Protection Mechanism

For the deduction of the storage fees we have a static value called **minTokenBalance** in the smart contract. This value is set to a reasonable threshold, so as to prevent small amounts of BWG tokens to lie in an address and generate storage fees, while being too low in value to be used for anything effectively.

That means that during the deduction of storage fees from a user, if the remaining balance after a deduction is less than the minimum threshold token balance, then we will deduct all available tokens from this wallet.

Remove Empty Wallet-Addresses From Storage Wallets List

StorageOperator has the ability to remove an empty wallet from the storage wallets list using the **BWGToken** contract. It could happen before or after the batch process. First of all, the on-chain server collects a list of the first available empty wallet batch from the **getEmptyWallets** method then sends it to **bulkRemoveEmptyWallets**.

Handling Costfree Wallets

The costfree wallet is a special type of wallet in which the tokens held there will not be considered for monthly storage cost deductions. The **maintainer** will be able to add or remove costfree wallets. The cost free wallets will only be used for internal wallets of the BrettonWoods *digital* AG, to prevent storage cost deductions from internal wallets and thus prevent unnecessary transaction fees. The Bretton Woods *digital* AG needs to settle the storage costs with the storage partners directly anyway and there is no need to collect any tokens form a company wallet for that.

Role Based Access Control

Access control provides a general role based access control mechanism. Multiple hierarchical roles can be created and assigned, each to multiple accounts. Each role can only perform the actions (execute functions) it was assigned to manage. A role can be granted to multiple wallets. An **admin** can grant or revoke other roles.

For example, for the **BWGToken** contract we have implemented two different roles:

One is called **admin** (set to gnosis safe multisig wallet for added security) and the other is called **maintainer**. The **admin** can grant or revoke any role to any wallet. The **admin** is a gnosis safe wallet which can perform the actions assigned to it, where it requires confirmation from multiple owners. Based on smart contracts different roles are created and assigned for different tasks.

- **BWGToken** contract has **admin** and **maintainer** roles.
- **StorageOperator** contract has **admin, maintainer** and **batch-executor** roles.
- **TokenTransferOperator** contract has **admin** and **maintainer** roles.
- **HotWalletTransferer** contract has **admin** and **maintainer** roles.

Admin role: An **admin** can grant or revoke other roles including **admin**. But can't revo- ke or renounce his own role. In general, **admin** could be a cold wallet or gnosis safe multisig wallet.

Maintainer role: The **maintainer** can add/update/remove some specific states of the smart contract. It could be an individual wallet.

Batch Executor role: The **batch-executor** can perform a monthly batch process. It should be an individual wallet so that an automated script can handle the batch pro- cess in a specific time interval.

The list of operations can be done by user role.

Role	Type	Grant Role	Renounce Role	Revoke Role	Smart Contract
Admin	Gnosis Safe Multisig Wallet	Admin, Maintainer and Batch Executor	– Other Admin – Maintainer and Batch Executor	No	BWGToken, StorageOperator, TokenTransferOperator HotWalletTransferer
Maintainer	Individual Wallet	No	No	Yes	BWGToken, StorageOperator, TokenTransferOperator HotWalletTransferer

Batch Executor	Individual Wallet	No	No	Yes	StorageOperator

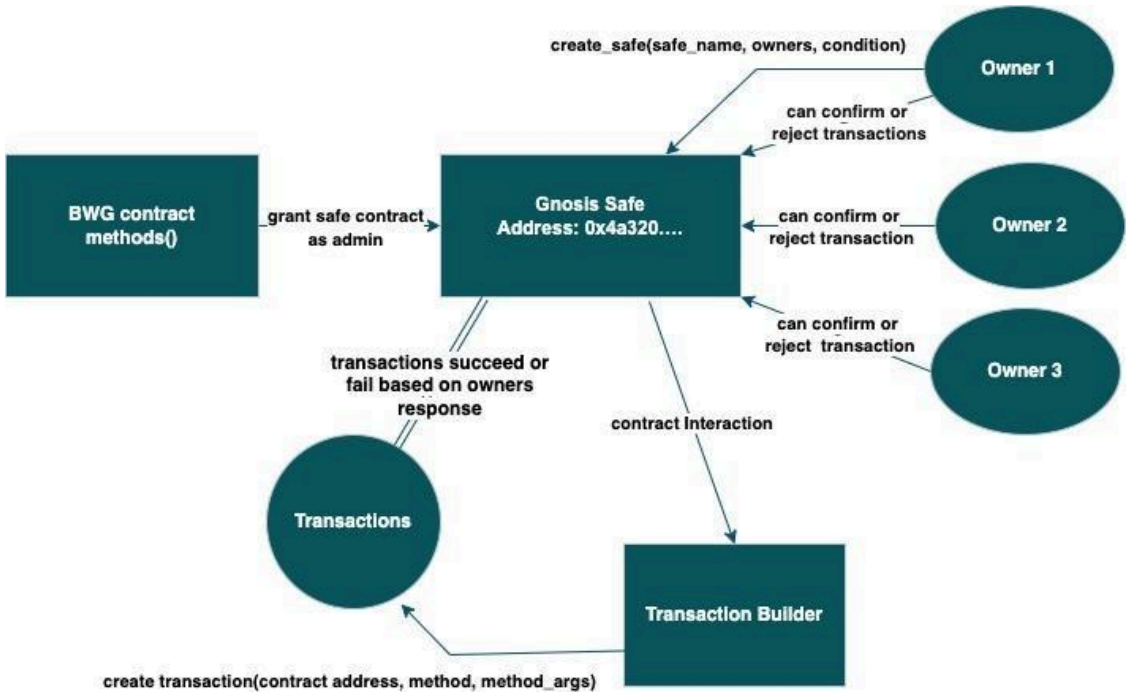
Gnosis Safe

Gnosis Safe is a smart contract wallet running on Ethereum that requires a minimum number of people to approve a transaction before it can occur.

How does it work?

The Gnosis Safe is a multi-signature smart contract wallet that allows users to define a list of owner/signer accounts and a threshold number of signers required to confirm a transaction. Once the threshold of owner accounts have confirmed a transaction, the Safe transaction can be executed.

Multisig (e.g. 2 of 3 owners) transaction



BWG Token Contract Overview

Method Name	Request Type	params	Role	Comments
allowance	Read	address owner, address spender	No	Returns the remaining number of tokens that spender will be allowed to spend on behalf of the owner through transferFrom. This is zero by default.
balanceOf	Read	address account	No	Returns the amount of tokens owned by the account.
decimals	Read		No	Returns the number of decimals used to get its user representation.
defaultOperators	Read		No	Returns the list of default operators. These accounts are operators for all token holders, even if authorizeOperator was never called on them.
granularity	Read		No	Returns the smallest part of the token that is not divisible. This means all token operations (creation, movement and destruction) must have amounts that are a multiple of this number
name	Read		No	Returns the name of the token.
symbol	Read		No	Returns the symbol of the token.
totalSupply	Read		No	Returns the totalSupply of the token.
costfreeWallets	Read		No (storageOperator)	Return list of costfreeWallets of the token.
storageWallets	Read		No (storageOperator)	Return storageWallets list by calling from storageOperator, otherwise empty list.
isCostableWallet	Read		No	Return true if the token holder is not in the costfree list otherwise false.
isCostfreeWallet	Read		No	Return true if wallet is added to costfreelist, otherwise false.
getBatchSize	Read		No	The external method is used to know the current batch-size.
getEmptyWallets	Read		No (storageOperator)	The getEmptyWallets method is used to find all available wallets with an empty balance.
addCostfreeWallet	Write	address wallet	No (storageOperator)	The method is used to add wallet to costfree list and emit an AddCostFreeWallet event. Emits an AddCostFreeWallet event.
removeCostfreeWallet	Write	address wallet	No (storageOperator)	The method is used to remove wallet from costfree list and emit RemoveCostFreeWallet event. Emits a RemoveCostFreeWallet event.

bulkRemoveEmptyWallets	Write	address [] wallet	No (storageOperator)	The method is used to remove zero balance wallets from storageWallet list.

Method Name	Request Type	params	Role	Comments
setStorageOperator	Write	address storage_operator_address	Maintainer	The method is used to update storage operator contract addresses so that valid contracts will get access to the storage wallet list.
Emits a StorageOperator event."	Read	address account	No	Returns the amount of tokens owned by the account.
setBatchSize	Write	uint256	Maintainer	The external method is used to update batch-size.
Emits a BatchSize event."	Read		No	Returns the list of default operators. These accounts are operators for all token holders, even if authorizeOperator was never called on them.
bulkSend	Write	address[] wallets, uint256[] costs	No	The method is used to send tokens to multiple wallets in a single transaction
pause	Write		No (storageOperator)	The method is used pause token transfer
Emits a Paused event.	Read		No	Returns the symbol of the token.
Requirements	Read		No	Returns the totalSupply of the token.
the contract must not be paused."	Read		No (storageOperator)	Return list of costfreeWallets of the token.
unpause	Write		No (storageOperator)	The method is used unpause token transfer
Emits a Unpaused event.	Read		No	Return true if the token holder is not in the costfree list otherwise false.
Requirements:	Read		No	Return true if wallet is added to costfreelist, otherwise false.
the contract must be paused."	Read		No	The external method is used to know the current batch-size.
burn	Write	uint256 amount, bytes data	No	Destroys amount tokens from the caller's account, reducing the total supply.
Emits a Burned event.	Write	address wallet	No (storageOperator)	The method is used to add wallet to costfree list and emit an AddCostFreeWallet event. Emits an AddCostFreeWallet event.
Requirements	Write	address wallet	No (storageOperator)	The method is used to remove wallet from costfree list and emit RemoveCostFreeWallet event. Emits a RemoveCostFreeWallet event.

the caller must have at least an amount of tokens."	Write	address [] wallet	No (storageOperator)	The method is used to remove zero balance wallets from storageWallet list.

Method Name	Request Type	params	Role	Comments
send	Write	address recipient, uint256 amount, bytes data	No	Moves amount of tokens from the caller's account to the recipient. Emits a Sent event. Requirements: the caller must have at least an amount of tokens. recipient cannot be the zero address.
operatorSend	Write	address sender, address recipient, uint256 amount, bytes data, bytes operatorData	No	Moves amount of tokens from sender to recipient. The caller must be an operator of the sender. Emits a Sent event. Requirements: sender cannot be the zero address. sender must have at least an amount of tokens.the caller must be an operator for the sender. recipient cannot be the zero address.
authorizeOperator	Write	address operator	No	Make an account an operator of the caller. Emits an AuthorizedOperator event. Requirements : operator cannot be calling address.
transfer	Write	address recipient, uint256 amount	No	Moves amount of tokens from the caller's account to the recipient. Returns a boolean value indicating whether the operation succeeded. Emits a Transfer event.
approve	Write	address spender, uint256 amount	No	Sets amount as the allowance of spender over the caller's tokens. Returns a boolean value indicating whether the operation succeeded.
transferFrom	Write	address sender, address recipient, uint256 amount	No	Moves amount of tokens from sender to recipient using the allowance mechanism. amount is then deducted from the caller's allowance. Returns a boolean value indicating whether the operation succeeded. Emits a Transfer event.

revokeRole	Write	bytes32 role, address account	Admin	Override Access control revokeRole allows them to revoke all roles except the admin.
renounceRole	Write	bytes32 role, address account	Admin	Override Access control renounceRole allows them to renounce all roles except the admin.

BWG Token Events

Custom events: StorageOperator(address wallet) AddCostFreeWallet(address wallet)

RemoveCostFreeWallet(address wallet)

RemoveCostableWallet(address wallet)

BatchSize(uint256 batchSize)

ERC-777 Events:

Sent(address operator, address from, address to, uint256 amount, bytes data, bytes operatorData)

Minted(address operator, address to, uint256 amount, bytes data, bytes operatorData) Burned(address operator, address from, uint256 amount, bytes data, bytes operatorData) AuthorizedOperator(address operator, address tokenHolder)

RevokedOperator(address operator, address tokenHolder)

ERC-20 Events:

Transfer(address from, address to, uint256 value) Approval(address owner, address spender, uint256 value)

Pausable

Paused(address account)

Unpaused(address account)

StorageOperator Contract Overview

Method Name	Request Type	params	Role	Comments
storageWalletCount	Read		No	Return the number of token holder wallets.
costFreeWallets	Read		No	Returns the list of token holder wallets that are not allowed to pay storage cost.
costFreeWalletCount	Read		No	Returns the number of token holder wallets that are not allowed to pay storage cost.
getBatchCursor	Read		No	The method used to get the lastIndex of the current batch cursor during the batch process.
getBatchSize	Read		No	The method used to get the batch size of the batch process.
getCostableWalletBatch	Read		No	The external method used to get all the information of the next batch. Returns: address[] wallets, uint256[] costs, uint256 totalCost, bool isCompleted **Valid batch request required.
getFundAddress	Read		No	Return updated funcaddress.
getLastPerformTimestamp	Read		No	Return last updated timestamp of batch process execution.
getMinTokenBalance	Read		No	Return last updated value of minTokenBalance .
getPerformDay	Read		No	Return day of the month of batch process.
getPerformTimeDuration	Read		No	Return interval of batch process. Like 1: Monthly 3: Quarterly 6: Half Yearly 12: Yearly
getStorageCostRate	Read		No	Return storageCostRate of batch process.
getTransactionFee	Read		No	Return average transaction fee of each user will be deducted during the batch process.

getEmptyWallets	Read		No	The external method used to get first available empty wallet array

Method Name	Request Type	params	Role	Comments
getStorageOverview	Read		No	Return all information regarding the upcoming batch process. uint256 totalStorageCost, uint256 totalWallet, uint256 totalWalletUnderThreshold, uint256 totalCostUnderThreshold, uint256 date, uint256 duration, uint256 lastPerformTime, uint256 costRate
pause	Read		Maintainer	The method is used to pause token transfers Emits a Paused event. Requirements: • the contract must not be paused.
unpause	Write		Maintainer	The method is used to unpause token transfers Emits an Unpaused event. Requirements: • the contract must be paused.
bulkRemoveEmptyWallets	Write	address[] calldata wallets	BatchExecutor	The external method used to remove from storage wallet. **Not allowed to perform during the batch process.
initBatchProcess	Write	uint256 walletMaxlimit	BatchExecutor	The external method used to initiate the batch process. Emits a InitBatchProcess event.**Valid batch request required
batchSend	Write	address[] calldata wallets, uint256[] calldata costs, uint256 currentIndex, uint256 totalCost	BatchExecutor	The external method used to perform batch process to collection storage fees Emits a StorageCostSummary event. **Valid batch request required

setStorageCostRate	Write	uint256 storageCostRate	Maintainer	The external method is used to set storageCostRate. The floating point number is not allowed to be set as rate so that rate is always 100 times of original rate. Emits a StorageCostRate event.
setTransactionFee	Write	uint256 transactionFee	Maintainer	The external method is used to update transactionFee. Emits a TransactionFee event.

Method Name	Request Type	params	Role	Comments
setFundAddress	Write	address fundAddress	Maintainer	The external method is used to update fundAddress, where the monthly storage cost is transferred. Emits a FundAddress event.
setLastPerform-Timestamp	Write	uint256 timestamp	Maintainer	The external method is used to update lastPerformTimestamp. • Maintainer permission required to change lastPerformTimestamp. • The newTimestamp should not be greater than the block timestamp. • Not allowed to perform during the batch process. Emits a LastPerformTimestamp event.
setMinTokenBalance	Write	uint256 minTokenBalance	Maintainer	The external method is used to update minTokenBalance. • Maintainer permission required to change minTokenBalance.. • Not allowed to perform during the batch process. Emits a MinTokenBalance event.
setPerformTime-Duration	Write	uint256 duration	Maintainer	The external method is used to update the interval of the batch process. Emits a PerformTimeDuration event.
setPerformDay	Write	uint256 dayOfMonth	Maintainer	The external method is used to update performDay (day of month). Emits a PerformDay event.
setBatchSize	Write	uint256 batchSize	Maintainer	The external method is used to set transaction batchSize. Emits a BatchSize event.
addCostFreeWallet	Write	address wallet	Maintainer	The addCostFreeWallet method is used to add a costfree wallet to the list. • Maintainer permission required to change minTokenBalance.. • Not allowed to perform during the batch process.
removeCost-FreeWallet	Write	address wallet	Maintainer	The removeCostFreeWallet method is used to remove a wallet from the costable wallet list. • Maintainer permission required to change minTokenBalance. • Not allowed to perform during the batch process.

revokeRole	Write	bytes32 role, address account	Admin	Override Access control revokeRole allows them to revoke all roles except the admin.
renounceRole	Write	bytes32 role, address account	Admin	Override Access control renounceRole allows them to renounce all roles except the admin.

StorageOperator Events

InitBatchProcess(uint256 maxWallet)

LastPerformTimestamp(uint256 timestamp)

MinTokenBalance(uint256 minBalance)

PerformTimeDuration(uint256 duration)

PerformDay(uint256 day) StorageCostRate(uint256

storageCostRate) FundAddress(address fundAddress)

BatchSize(uint256 batchSize) TransactionFee(uint256

fee)

StorageCostSummary(address fundAddress, uint256 storageCostRate, uint256 totalFee)

Method Name	Request Type	params	Role	Comments
getOperator	Read		No	Returns the storage cost operator contract address.
setOperator	Write	address operator	Yes	The external method is used to set the operator. Emits an Operator event.
operatorSend	Write	IERC-777 token, address from, address to, uint256 amount	Yes	The operatorSend used to transfer tokens on behalf of token holders.
revokeRole	Write	bytes32 role, address account	Admin	Override Access control revokeRole allows them to revoke all roles except the admin.
renounceRole	Write	bytes32 role, address account	Admin	Override Access control renounceRole allows them to renounce all roles except the admin.

TokenTransferOperator Events

Operator(address operator)

HotWalletTransferer Contract Overview

Method Name	Request Type	params	Role	Comments
checkUpkeep	Read		No	The checkUpkeep is an external KeeperCompatibleInterface method used to check if upkeepNeeded is needed.
performUpkeep	Write		No	The performUpkeep is an external KeeperCompatibleInterface method used to run performUpkeep if it gets true from the checkUpkeep method. Emits an TransferToHotWallet event.
setHolderAddress	Write	address newHolder	Maintainer	The external method is used to update the holder. Emits an HolderAddress event.
setHotWalletAddress	Write	address hotwallet	Maintainer	The external method is used to update the hotwallet address. Emits an HotWalletAddress event.
revokeRole	Write	bytes32 role, address account	Admin	Override Access control revokeRole allows them to revoke all roles except the admin.
renounceRole	Write	bytes32 role, address account	Admin	Override Access control renounceRole allows them to renounce all roles except the admin.

HotWalletTransferer Events

TransferToHotWallet(address indexed holder, address indexed hotWallet, uint256 value)

HotWalletAddress(address indexed hotWalletAddress)

HolderAddress(address indexed holder) TriggerTokensAmount(uint256

triggerTokensAmount) TokensAmountToAdd(uint256

tokensAmountToAdd)

Coverage Reports

File	Stmts%	Branch	Func%	Line%	Uncovered Lines
Contracts	**100**	**97.41**	**100**	**100**	
BWGToken.sol	100	95.83	100	100	
HotWalletTransferer.sol	100	100	100	100	
StorageOperator.sol	100	97.73	100	100	
TokenTransferOperator.sol	100	100	100	100	

** Please use node version v16.13.0 to get coverage reports. It doesn't work with the latest node version.

Disclaimer

This document constitutes non-binding preliminary information which is intended solely for advertising purposes and is not a prospectus within the meaning of the European Securities Prospectus Act, the German Investment Act or the German Capital Investment Code or any corresponding foreign law. This presentation is neither an offer nor a solicitation of an offer to purchase securities. The information in this presentation does not constitute investment advice or an investment recommendation. While every care has been taken in the preparation of this presentation, errors and omissions excepted. The statements made are based on estimates, economic data, own assessments and are forward-looking statements at the time of preparation of the presentation and may be subject to change. Persons into whose possession this presentation comes are required to inform themselves about and to observe all appli- cable laws and regulations.