Track & Trace

Track & Trace APIs provide the solution for managing decentralized records for end-to-end tracking & tracing of a product in shipment. These APIs can be integrated with existing blockchain products or products that would require a decentralized trustless method to trace their product's journey. These APIs create an asset on the smart contract with a unique identifier that is transferred down the supply chain managing trail of every action performed on it, and ownership is transferred to all the participants with them signing the transactions.
A single unit can be packaged into a carton/container with its unique id and then can be inwarded/outwarded and all the units in it can be transferred to new owner all at once, afterwards, they can be unpackaged for micro transfers.

Free Users: The smart contracts for these API's have been deployed on the Goerli Testnet, allowing free users to test and interact with the API's functionality.

Premium Users: Premium users, who have purchased our plans, will have the option to select their own blockchain network for deploying their separate smart contract. Infrablok supports the following EVM compatible blockchain networks :

1. Ethereum
2. Polygon
3. Avalanche
4. BNB
5. Telos
6. Goerli Testnet
7. Polygon Testnet
8. Avalanche Testnet
9. BNB Testnet
10. Telos Testnet

Smart Contract Deployment:

To deploy your Separate smart contract on desired chain, you need to complete these steps:

1. Select Desired Chain
2. Create Wallet**
3. Topup/transfer native coin of your desired chain
4. Deploy Contract

** We will create a wallet address on your behalf. We will keep the private key in encrypted form in our database. Gas fees will be deducted from this wallet when we deploy your contract on the desired chain and when you use our API's. You are strongly advised to use this wallet address for "Track n Trace" API's only.

List of Supply Chain APIs

1. Create a new asset

API: https://trackntrace.infrablok.com/api/asset/createAsset

This process involves registering products, a single unit at a time.

Input:

```
{

"productid": "string",

"metadata": {

},

"owner": "string",

"latitude": "string",

"longitude": "string",

"comment": "string"

}
```

Below are the details required to register a single product:

1. **Productid:** It is the product ID, can be autogenerated at our side or can be provided from caller's ERP system.
2. **Metadata:** It is the product metadata details in JSON format, e.g. {*model: "IVHQE2022", SKU: "SKU00123", …*}
3. **Owner:** It is the wallet address (EVM) of the user, to whom the ownership of the product should be assigned at the time of registration of the product.
4. **Latitude & Longitude:** It is the initial location of the product, where it was manufactured.
5. **Comment:** It is any comment/description provided by the user.

2. Create a new package

API: https://trackntrace.infrablok.com/api/asset/createPackage

This process involves packaging a list of products together into one package.

Input:

```
{

"packageId": "string",

"productIdList": [

"string"

]

}
```

Below are the details required to create a package:

1. **Packageid:** It is the package id.
2. **ProductList:** It is the array of product (asset) ids need to be packaged together.

3. Create an outward entry

API: https://trackntrace.infrablok.com/api/asset/createOutwardEntry

This process involves out warding the group of products or group of packages from product owner to logistics provider which involves changing ownership of all the products either provided as single product list or package list to logistics provider.

Input:

```
{

"type": "string",

"IdList": [

"string",”string”,….,” string”

],

"receiverAdd": "string",

"logisticAdd": "string",

"invoiceHash": "string",
```

```
"invoiceNum": "string",

"owner": "string",

"latitude": "string",

"longitude": "string",

"comment": "string"

}
```

Below are the details required to create an outward:

1. **Type:** It is to state if the provided id list is of type UNIT or PACKAGE.
2. **IdList:** It is the array of product ids or package ids.
3. **ReceiverAdd:** It is the wallet address (EVM) of the Buyer.
4. **LogisticAdd:** It is the wallet address (EVM) of the Logistic Provider, to whom the ownership of the product should be assigned at the time of outwarding.
5. **InvoiceHash:** It is the hash of the invoice of the dispatched products.
6. **InvoiceNum:** It is the invoice number of the dispatched products.
7. **Owner:** It is the wallet address (EVM) of the Seller, who has the ownership of the product.
8. **Latitude & Longitude:** It is the current location of the product.
9. **Comment:** It is any comment/description provided by the seller.

4. Create an inward entry

API: https://trackntrace.infrablok.com/api/asset/createInwardEntry

In this process the receiver provides the group of products ids or the group of package ids that are received. It involves changing ownership of all the products either provided as a sole product list or package list to the receiver.

Input:

{

"type": "string",

"IdList": [

"string",”string”,….,” string”

],

"receiverAdd": "string",

"logisticAdd": "string",

"invoiceHash": "string",

"invoiceNum": "string",

"latitude": "string",

"longitude": "string",

"comment": "string"

```
}
```

Below are the details required to create an inward:

1. **Type:** It is to state if the provided id list is of type UNIT or PACKAGE.
2. **IdList:** It is the array of product ids or package ids.
3. **ReceiverAdd:** It is the wallet address (EVM) of the Buyer, to whom the ownership of the product should be assigned at the time of inwarding.
4. **LogisticAdd:** It is the wallet address (EVM) of the Logistic Provider.
5. **InvoiceHash:** It is the hash of the invoice of the dispatched products.
6. **InvoiceNum:** It is the invoice number of the dispatched products.
7. **Latitude & Longitude:** It is the current location of the product.
8. **Comment:** It is any comment/description provided by the buyer.

5. Product delivery to end user (Setting status as SOLD)

API: https://trackntrace.infrablok.com/api/asset/assetSold

In this process the buyer provides the group of products ids or the group of package ids that the end user received. It involves changing ownership of all the products either provided as a sole product list or package list to end user and setting final state to SOLD.

Input:

```
{
```

```
"type": "string",
```

```
"IdList": [
```

```
"string", "string",..., "string"
```

```
],

"receiverAdd": "string",

"invoiceHash": "string",

"invoiceNum": "string",

"owner": "string",

"latitude": "string",

"longitude": "string",

"comment": "string"

}
```

Below are the details required:

1. **Type:** It is to state if the provided id list is of type UNIT or PACKAGE.
2. **IdList:** It is the array of product ids or package ids.
3. **ReceiverAdd:** It is the wallet address (EVM) of the End User, to whom the ownership of the product should be assigned.
4. **InvoiceHash:** It is the hash of the invoice of the dispatched products.
5. **InvoiceNum:** It is the invoice number of the dispatched products.
6. **Owner:** It is the wallet address (EVM) of the Buyer, who has the ownership of the product.
7. **Latitude & Longitude:** It is the current location of the product.
8. **Comment:** It is any comment/description provided by the buyer.

6. Depackage a package

API: https://trackntrace.infrablok.com/api/asset/dePackage

This process involves depackaging a package. Here assets packed within the provided package id will be unpacked again.

Input:

{

"packageId": "string"

}

1. **Packageid:** It is the package id.

7. Track an asset

API: https://trackntrace.infrablok.com/api/asset/productTraceById

This process returns all the tracking details of an asset.

Input:

Key: productid Value: "string"

Productid: It is the product id. For e.g., “101”.

8. Get an asset detail

API: https://trackntrace.infrablok.com/api/asset/getAssetDetailsById

This process gives details about an asset.

Input:

Key: productid Value: "string"

Productid: It is the product id. For e.g., “101”.

9. Check if an asset exists for a user

API: https://trackntrace.infrablok.com/api/asset/assetExistsByUserAddress

This process checks if an asset exists for the provided user address and returns “true” or “false” in response to the call.

Input:

Key: address **Value:** "string"

Key: productid **Value:** "string"

Address: It is the wallet address (EVM) of the user. For e.g., 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

Productid: It is the product id. For e.g., “101”.

10. Get all asset ids

API: https://trackntrace.infrablok.com/api/asset/getAllAssets

This process returns an array of asset ids registered on blockchain so far.

11. Get all package ids

API: https://trackntrace.infrablok.com/api/asset/getAllPackages

This process returns an array of package ids registered on blockchain so far.

12. Get package detail

API: https://trackntrace.infrablok.com/api/asset/getAllProductByPackageId

This process returns an array of asset ids mapped with the given package id.

Input:

Key: productid Value: "string"

Productid: It is the product id. For e.g., “101”.

13. Transfer Contract Ownership

API: https://trackntrace.infrablok.com/api/asset/transferContractOwnership

This function is used to change contract owner. Only the current contract owner can make this call.

Input:

```
{


"newOwner ": "string"


}
```

1. **NewOwner:** It is the wallet address of the new contract owner.

14. Check Contract Owner

API: https://trackntrace.infrablok.com/api/asset/checkContractOwner

This function is used to check the owner of the contract.

Service Level Agreement

Overview

SLA stands for service level agreement, which is an agreement signed between two parties, logistic and sender of a goods, which contains the rules and regulations and penalty criteria in case of violations, and logistic entity is liable to pay the penalty automatically with the help of the smart contract, which is holding all the transit data, like temperature, humidity, vibrations etc.

List of SLA APIs

1. Deposit Fund

API: https://trackntrace.infrablok.com/api/asset/depositFund

This method is used to fund smart contract. An amount greater than 0 is required to be sent as **msg.value** along with function call. The value is stored in balance variable which can later be used to fetch contract balance. The amount should be provided in **WEI**.

Input:

```
{

"amount": int

 }
```

2. Set Logistic Charge

API: https://trackntrace.infrablok.com/api/asset/setLogisticCharge

This method is used to set charges of logistic providers for a given consignment. It takes **logisticProvider** (a EVM wallet address of logistic provider), **invoiceNumber** and **charge** as input.

Input:

```
{
"logisticProvider":"string", "invoiceNumber": "string", "charge":int
 }
```

3. Set Metric

API: https://trackntrace.infrablok.com/api/asset/setMetric

This method is used to set metrics for a consignment. It takes **invoiceNumber (invoice number of consignment)** and a **tuple** of type **Metric** as input. It sets one metric at a time where each metric is mapped

with the invoiceNumber and an index (starting with 0). Metric stands for criteria I.e., temperature, pressure etc. and their max-min range in which the goods are to be kept in, these metrics will be checked with the data obtained from the sensers.

struct Metric {

string name;

uint256 minRange;

uint256 maxRange;

}

Where, **name** is name of a metric, **minRange** is minimum value of a metric and **maxRange** is maximum value of a metric.

Input:

```
{
"invoiceNumber": "string",
```

```
 "metricName": "string",

 "minRange":int,

 "maxRange":int

}
```

4. Set Penalty

API: https://trackntrace.infrablok.com/api/asset/setPenalty

This method is used to set penalties for a consignment if or when a metric value goes out of range. It takes invoiceNumber , metricID(metric index) and an array of tuples of type Penalty as input. It sets range and charge for one metric at a time where each array is mapped with the invoiceNumber and metricID(metric index).

struct Penalty {

uint16 minCount;

uint16 maxCount;

uint256 charge;

}

Where, **minCount** is minimum violation count, **maxCount** is maximum violation count and **charge** is penalty fee.

```
Input:

{



"invoiceNumber":"string",



"metricID":int,



"penalty":[
```

```
{

"minCount": int,

"maxCount": int,

"charge": int

},

{

"minCount": int,

"maxCount": int,

"charge": int

}

]

}
```

5. Set Metric Value

API: https://trackntrace.infrablok.com/api/asset/setMetricValue

This method is used to update metric values at regular time intervals. It sets one value at a time. Here each metric value is mapped with the **invoiceNumber**, **metricID(metric index)** and the index(indexes for values starting with 0)

```
Input:

{

 "invoiceNumber": "string",

 "metricID":int,

 "metricValue":int

 }
```

6. Calculate Penalty

API: https://trackntrace.infrablok.com/api/asset/calculatePenalty

This method is used to calculate total penalties based on total number of violations count for a consignment. It takes **invoiceNumber** and an array of **metricIDs(metric indexes)**.

```
Input:

{

"invoiceNumber":"string",

 "metricID":[int,int]

 }
```

7. Settlement

API: https://trackntrace.infrablok.com/api/asset/Settlement

This method is used for settlement. It takes **invoiceNumber,an array of metricIDs** (metric indexes) and **logisticProvider address** (a EVM wallet address of the logistic provider) as input and penalty is calculated. With this method call penalty

amount is deducted from the charge set for the logistic provider and the remaining amount is transferred to the logistic provider from the smart contract.

Input:

```
{
"invoiceNumber":"string",
"metricID":[int,int],
"logisticProvider":"string"
 }
```

***By default, solidity creates getter functions for public variables. Using web3 we can directly get the results using the variable name.**

Below are the public variables and mappings used in the contract to store and fetch the results.

8. Check Contract Balance

API: https://trackntrace.infrablok.com/api/asset/getContractBalance

This method is used to fetch contract balance.

9. Consignment Metric Map

API: https://trackntrace.infrablok.com/api/asset/getMetricByMetricId

This mapping returns a metric stored at the given index for a given consignment.

Input:

Key: invoiceNumber **Value:** "string"

Key: metricID **Value:** int

InvoiceNumber: It is the invoice number of assets/packages outwared.

MetricID: It is the metric index, starting with 0.

10.Counter Map

API: https://trackntrace.infrablok.com/api/asset/getTotalMetricByInvoice

This mapping returns total metric stored for a given consignment.

Input:

Key: invoiceNumber **Value:** "string"

Key: metricID **Value:** int

Key: index **Value:** int

Index: It is the metric value index, starting with 0.

11. Metric Value Map

API: https://trackntrace.infrablok.com/api/asset/getMetricValueByMetricId

This mapping returns metric value stored at an index for given metric of given consignment.

Input:

Key: invoiceNumber **Value:** "string"

Key: metricID **Value:** int

Key: index **Value:** int

Index: It is the metric value index, starting with 0.

12. Value Count Map

API: https://trackntrace.infrablok.com/api/asset/getTotalMetricValueByMetricId

This mapping returns total metric values stored for a given metric id for a given consignment.

Input:

Key: invoiceNumber **Value:** "string"

Key: metricID **Value:** int

13. Penalty Map

API: https://trackntrace.infrablok.com/api/asset/getPenaltyByMetricId

This mapping returns the penalty range and charge (struct) stored at an index for the given metric of a given consignment.

Input:

Key: invoiceNumber **Value:** "string"

Key: metricID **Value:** int

Key: index **Value:** int

Index: It is the penalty object index, starting with 0.

14. Penalty Updated

API: https://trackntrace.infrablok.com/api/asset/isPenaltyUpdated

This mapping is used to check if a penalty has been set for a given invoice number and metric id. This returns true or false in response.

Input:

Key: invoiceNumber **Value:** "string"

Key: metricID **Value:** int

15. Settled

API: https://trackntrace.infrablok.com/api/asset/isSettled

This mapping is used to check if settlement has been made for a given consignment. This returns true or false in response.

Input:

Key: logisticProvider **Value:** "string"

Key: invoiceNumber **Value:** "string"

LogisticProvider : It is the wallet address of logistic provider.

16. Violation Per Metric

API: https://trackntrace.infrablok.com/api/asset/getViolationCountByMetricId

This mapping returns the total violation count for a given invoice number and metric id.

Input:

Key: invoiceNumber **Value:** "string"

Key: metricID **Value:** int

17. Violation

API: https://trackntrace.infrablok.com/api/asset/getViolationCountByInvoice

This mapping returns the total violation count for a given consignment.

Input:

Key: invoiceNumber **Value:** "string"

18. Logistic Charge

API: https://trackntrace.infrablok.com/api/asset/getLogisticCharge

This mapping returns the charge set for a logistic provider for the given invoice number.

Input:

Key: logisticProvider **Value:** "string"

Key: invoiceNumber **Value:** "string"