GetUserLinkedAccount

SDK Type: *Javascript/Node.js*

Description Gets an Asset Issuer account for a User.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.getUserLinkedAccount(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

GetUserInventoryWithPayload

SDK Type: *Javascript/Node.js*

Description Gets a User's inventory with payload data.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.getUserInventoryWithPayload(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateMarketplaceAssetUnlimited

SDK Type: *Javascript/Node.js*

Description Creates an Asset from an Asset Schema.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*
- **json_payload** ~ *The JSON schema payload, which is required to mint an Asset.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
var json_payload = "";
sdk.createMarketplaceAssetUnlimited(asset_schema_uid, json_payload).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateMarketplaceAssetUnlimitedWithImage

SDK Type: *Javascript/Node.js*

Description Creates an Asset from an Asset Schema.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*
- **json_payload** ~ *The JSON schema payload, which is required to mint an Asset.*
- **image_file** ~ *The directory location of an image file.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
var json_payload = "";
var image_file = "";
sdk.createMarketplaceAssetUnlimitedWithImage(asset_schema_uid, json_payload, image_file).then((data)
=> {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateMarketplaceAssetQuantity

SDK Type: *Javascript/Node.js*

Description Creates an Asset from an Asset Schema.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*
- **json_payload** ~ *The JSON schema payload, which is required to mint an Asset.*
- **quantity** ~ *The number of digital assets listed for sale.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
var json_payload = "";
var quantity = "";
sdk.createMarketplaceAssetQuantity(asset_schema_uid, json_payload, quantity).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateMarketplaceAssetQuantitydWithImage

SDK Type: *Javascript/Node.js*

Description Creates an Asset from an Asset Schema.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*
- **json_payload** ~ *The JSON schema payload, which is required to mint an Asset.*
- **image_file** ~ *The directory location of an image file.*
- **quantity** ~ *The number of digital assets listed for sale.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
var json_payload = "";
var image_file = "";
var quantity = "";
sdk.createMarketplaceAssetQuantitydWithImage(asset_schema_uid, json_payload, image_file,
quantity).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateMarketplaceListing

SDK Type: *Javascript/Node.js*

Description Assigns an Asset to be listed on the marketplace.

Required Inputs

- **user_inventory_uid** ~ *The UID of an Asset in a User's inventory*
- **sale_price** ~ *The Price set in cents (e.g., 100 = 1.00 USD)*
- **listing_data** ~ *Used to display additional data for an Asset listed on sale.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**
4. **CreateMarketplaceAssetQuantity**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var user_inventory_uid = "";
var sale_price = "";
var listing_data = "";
sdk.createMarketplaceListing(user_inventory_uid, sale_price, listing_data).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateUserAssetIssuerOwnership

SDK Type: *Javascript/Node.js*

Description Links ownership of an existing User to an Asset Issuer.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*
- **game_username** ~ *The username from an Asset Issuer's database to link to a User.*
- **game_uid** ~ *The Unique Identifier from an Asset Issuer's database to link a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
var game_username = "";
var game_uid = "";
sdk.createUserAssetIssuerOwnership(asset_issuer_uid, user_uid, game_username, game_uid).then((data)
=> {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UpdateUserAssetIssuerOwnership

SDK Type: *Javascript/Node.js*

Description Updates ownership of an existing User to an Asset Issuer.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*
- **game_username** ~ *The username from an Asset Issuer's database to link to a User.*
- **game_uid** ~ *The Unique Identifier from an Asset Issuer's database to link a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
var game_username = "";
var game_uid = "";
sdk.updateUserAssetIssuerOwnership(asset_issuer_uid, user_uid, game_username, game_uid).then((data)
=> {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

DeleteUserAssetIssuerOwnership

SDK Type: *Javascript/Node.js*

Description Removes the ownership link between an Asset Issuer & a User.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.deleteUserAssetIssuerOwnership(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

DispatchExternalInvite

SDK Type: *Javascript/Node.js*

Description Invites Users to join the marketplace.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **game_email** ~ *The email address of a User.*
- **game_username** ~ *The username from an Asset Issuer's database to link to a User.*
- **game_uid** ~ *The Unique Identifier from an Asset Issuer's database to link a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var game_email = "";
var game_username = "";
var game_uid = "";
sdk.dispatchExternalInvite(asset_issuer_uid, game_email, game_username, game_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

ConsumeExternalInvite

SDK Type: *Javascript/Node.js*

Description Activates keys for a User.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **invite_key** ~ *The invite key sent to a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var invite_key = "";
sdk.consumeExternalInvite(asset_issuer_uid, invite_key).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateUserAssetIssuerAccessGrant

SDK Type: *Javascript/Node.js*

Description Grants access to inventory outside of a Game Session.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.createUserAssetIssuerAccessGrant(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

DeleteUserAssetIssuerAccessGrant

SDK Type: *Javascript/Node.js*

Description Deletes access to inventory outside of a Game Session.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.deleteUserAssetIssuerAccessGrant(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

AssignItem

SDK Type: *Javascript/Node.js*

Description Assigns a minted Asset to a User that has not been previously assigned.

Required Inputs

- **asset_uid** ~ *An Asset that has been minted from an Asset Schema, but has not been assigned to a User.*
- **recipient_uid** ~ *The User UID to receive an Asset.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_uid = "";
var recipient_uid = "";
sdk.assignItem(asset_uid, recipient_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

LoginUser

SDK Type: *Javascript/Node.js*

Description Enables User log in.

Required Inputs

- **user_email** ~ *A User's email address.*
- **user_password** ~ *A User's password.*

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var user_email = "";
var user_password = "";
sdk.loginUser(user_email, user_password).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UnlockUserInventory

SDK Type: *Javascript/Node.js*

Description Unlocks a User's inventory during a Game Session if the Game Session is locked.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **GrantUserOwnership**
4. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.unlockUserInventory(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UnlockUserInventoryOAuth

SDK Type: *Javascript/Node.js*

Description Unlocks a User's inventory during a Game Session if the Game Session is locked.

Required Inputs

- **oauth_access_token** ~ *The token that is obtained via 0Auth2.*
- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **GrantUserOwnership**
4. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var oauth_access_token = "";
var asset_issuer_uid = "";
var user_uid = "";
sdk.unlockUserInventoryOAuth(oauth_access_token, asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

LockUserInventory

SDK Type: *Javascript/Node.js*

Description Locks a User's inventory during a Game Session if the Game Session is unlocked.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **GrantUserOwnership**
4. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.lockUserInventory(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

LockUserInventoryOAuth

SDK Type: *Javascript/Node.js*

Description Locks a User's inventory during a Game Session if the Game Session is unlocked.

Required Inputs

- **oauth_access_token** ~ *The token that is obtained via 0Auth2.*
- **asset_issuer_uid** ~ *The UID of an Asset Issuer, which is defined as an entity that creates, issues, & manages digital assets.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **GrantUserOwnership**
4. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var oauth_access_token = "";
var asset_issuer_uid = "";
var user_uid = "";
sdk.lockUserInventoryOAuth(oauth_access_token, asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

MintAssetSchemaForUser

SDK Type: *Javascript/Node.js*

Description Enables an Asset Issuer to mint an Asset Schema into an Asset & then assign it to a User's inventory.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*
- **user_uid** ~ *The UID of a User.*
- **json_payload** ~ *The JSON schema payload, which is required to mint an Asset.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
var user_uid = "";
var json_payload = "";
sdk.mintAssetSchemaForUser(asset_schema_uid, user_uid, json_payload).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UpdateInventoryAsset

SDK Type: *Javascript/Node.js*

Description Allows an Asset Issuer to update an Asset within a User's inventory.

Required Inputs

- **inventory_uid** ~ *The UID of an Asset stored on a User's account.*
- **json_payload** ~ *The JSON schema payload, which is required to mint an Asset.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **MintAssetSchemaForUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var inventory_uid = "";
var json_payload = "";
sdk.updateInventoryAsset(inventory_uid, json_payload).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

GetUserInventoryAsset

SDK Type: *Javascript/Node.js*

Description Gets Asset data for an Asset within a User's inventory.

Required Inputs

- **asset_uid** ~ *An Asset within a User's inventory.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_uid = "";
sdk.getUserInventoryAsset(asset_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

GetAssetIssuer

SDK Type: *Javascript/Node.js*

Description Gets an Asset Issuer's data.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
sdk.getAssetIssuer(asset_issuer_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

GetAllAssetSchemas

SDK Type: *Javascript/Node.js*

Description Gets all Asset Schemas created by an Asset Issuer.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
sdk.getAllAssetSchemas(asset_issuer_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

GetAssetSchema

SDK Type: *Javascript/Node.js*

Description Gets a designated Asset Schema created by an Asset Issuer.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
sdk.getAssetSchema(asset_schema_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateAssetSchema

SDK Type: *Javascript/Node.js*

Description Creates an Asset Schema.

Required Inputs

- **schema_name** ~ *An Asset Schema's name.*
- **schema_json** ~ *A JSON Schema, which is required to create an Asset Schema* https://json-schema.org/
- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **isactive** ~ *Enables Asset Schemas to be minted; if "Inactive", the developer is unable to mint new Assets.*
- **istradable** ~ *Enables minted Assets to be traded between Users.*
- **issellable** ~ *Enables minted Assets to be sold.*
- **isrevocable** ~ *Enables an Asset Issuer to remove a minted Asset from a User's inventory.*
- **isburnable** ~ *Enables an Asset Issuer to delete a minted Asset from a User's inventory.*
- **tags** ~ *List of tags for an associated Asset Schema; Max is 5.*
- **category** ~ *Category ID for a category.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var schema_name = "";
var schema_json = "";
var asset_issuer_uid = "";
var isactive = false;
var istradable = false;
var issellable = false;
var isrevocable = false;
var isburnable = false;
var tags = "";
var category = "";
sdk.createAssetSchema(schema_name, schema_json, asset_issuer_uid, isactive, istradable, issellable,
isrevocable, isburnable, tags, category).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateAssetSchemaWithImage

SDK Type: *Javascript/Node.js*

Description Creates an Asset Schema.

Required Inputs

- **schema_name** ~ *An Asset Schema's name.*
- **schema_json** ~ *A JSON Schema, which is required to create an Asset Schema* https://json-schema.org/
- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **isactive** ~ *Enables Asset Schemas to be minted; if "Inactive", the developer is unable to mint new Assets.*
- **istradable** ~ *Enables minted Assets to be traded between Users.*
- **issellable** ~ *Enables minted Assets to be sold.*
- **isrevocable** ~ *Enables an Asset Issuer to remove a minted Asset from a User's inventory.*
- **isburnable** ~ *Enables an Asset Issuer to delete a minted Asset from a User's inventory.*
- **tags** ~ *List of tags for an associated Asset Schema; Max is 5.*
- **category** ~ *Category ID for a category.*
- **image_file** ~ *The directory location of an image file.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var schema_name = "";
var schema_json = "";
var asset_issuer_uid = "";
var isactive = false;
var istradable = false;
var issellable = false;
var isrevocable = false;
var isburnable = false;
var tags = "";
var category = "";
var image_file = "";
sdk.createAssetSchemaWithImage(schema_name, schema_json, asset_issuer_uid, isactive, istradable,
issellable, isrevocable, isburnable, tags, category, image_file).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

GetUserFromGameSession

SDK Type: *Javascript/Node.js*

Description Gets a User's data from a Game Session.

Required Inputs

- **user_session** ~ *The User Session passed from the Desktop Login App.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var user_session = "";
sdk.getUserFromGameSession(user_session).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateGameSession

SDK Type: *Javascript/Node.js*

Description Creates a Game Session for a User.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.createGameSession(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateGameSessionOAuth

SDK Type: *Javascript/Node.js*

Description Creates a Game Session for a User.

Required Inputs

- **oauth_access_token** ~ *The token that is obtained via 0Auth2.*
- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var oauth_access_token = "";
var asset_issuer_uid = "";
var user_uid = "";
sdk.createGameSessionOAuth(oauth_access_token, asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateLockedGameSession

SDK Type: *Javascript/Node.js*

Description Creates a Game Session with the inventory locked by an Asset Issuer for a User.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.createLockedGameSession(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateLockedGameSessionOAuth

SDK Type: *Javascript/Node.js*

Description Creates a Game Session with the inventory locked by an Asset Issuer for a User.

Required Inputs

- **oauth_access_token** ~ *The token that is obtained via 0Auth2.*
- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var oauth_access_token = "";
var asset_issuer_uid = "";
var user_uid = "";
sdk.createLockedGameSessionOAuth(oauth_access_token, asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

EndGameSession

SDK Type: *Javascript/Node.js*

Description Ends a Game Session for a User.

Required Inputs

- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var user_uid = "";
sdk.endGameSession(user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

EndGameSessionOAuth

SDK Type: *Javascript/Node.js*

Description Ends a Game Session for a User.

Required Inputs

- **oauth_access_token** ~ *The token that is obtained via 0Auth2.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var oauth_access_token = "";
var user_uid = "";
sdk.endGameSessionOAuth(oauth_access_token, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

LinkAssetIssuerAccountToUser

SDK Type: *Javascript/Node.js*

Description Links an Asset Issuer's account to a User account.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **user_uid** ~ *The UID of a User.*
- **game_username** ~ *The username from an Asset Issuer's database to link to a User.*
- **game_uid** ~ *The Unique Identifier from an Asset Issuer's database to link a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
var game_username = "";
var game_uid = "";
sdk.linkAssetIssuerAccountToUser(asset_issuer_uid, user_uid, game_username, game_uid).then((data) =>
{
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UnlinkAssetIssuerAccountFromUser

SDK Type: *Javascript/Node.js*

Description Unlinks an Asset Issuer's account to a User account.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **user_uid** ~ *The UID of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var user_uid = "";
sdk.unlinkAssetIssuerAccountFromUser(asset_issuer_uid, user_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UpdateAssetIssuer

SDK Type: *Javascript/Node.js*

Description Updates an Asset Issuer.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **issuer_name** ~ *An Asset Issuer's name.*
- **issuer_industry** ~ *An Asset Issuer's industry.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var issuer_name = "";
var issuer_industry = "";
sdk.updateAssetIssuer(asset_issuer_uid, issuer_name, issuer_industry).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateAssetIssuer

SDK Type: *Javascript/Node.js*

Description Creates an Asset Issuer.

Required Inputs

- **organization_uid** ~ *The UID of an Organization.*
- **issuer_name** ~ *An Asset Issuer's name.*
- **issuer_industry** ~ *An Asset Issuer's industry.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var organization_uid = "";
var issuer_name = "";
var issuer_industry = "";
sdk.createAssetIssuer(organization_uid, issuer_name, issuer_industry).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UpdateAssetIssuerWithImage

SDK Type: *Javascript/Node.js*

Description Updates an Asset Issuer with an image.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **issuer_name** ~ *An Asset Issuer's name.*
- **issuer_industry** ~ *An Asset Issuer's industry.*
- **image_file** ~ *The directory location of an image file.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var issuer_name = "";
var issuer_industry = "";
var image_file = "";
sdk.updateAssetIssuerWithImage(asset_issuer_uid, issuer_name, issuer_industry,
image_file).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateAssetIssuerWithImage

SDK Type: *Javascript/Node.js*

Description Creates an Asset Issuer with an image.

Required Inputs

- **organization_uid** ~ *The UID of an Organization.*
- **issuer_name** ~ *An Asset Issuer's name.*
- **issuer_industry** ~ *An Asset Issuer's industry.*
- **image_file** ~ *The directory location of an image file.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var organization_uid = "";
var issuer_name = "";
var issuer_industry = "";
var image_file = "";
sdk.createAssetIssuerWithImage(organization_uid, issuer_name, issuer_industry,
image_file).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UpdateAssetSchemaWithImage

SDK Type: *Javascript/Node.js*

Description Updates an existing Asset Issuer.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*
- **schema_name** ~ *An Asset Schema's name.*
- **schema_json** ~ *A JSON Schema, which is required to create an Asset Schema* https://json-schema.org/
- **issellable** ~ *Enables minted Assets to be sold.*
- **istradable** ~ *Enables minted Assets to be traded between Users.*
- **isrevocable** ~ *Enables an Asset Issuer to remove a minted Asset from a User's inventory.*
- **isburnable** ~ *Enables an Asset Issuer to delete a minted Asset from a User's inventory.*
- **isactive** ~ *Enables Asset Schemas to be minted; if "Inactive", the developer is unable to mint new Assets.*
- **tags** ~ *List of tags for an associated Asset Schema; Max is 5.*
- **category** ~ *Category ID for a category.*
- **image_file** ~ *The directory location of an image file.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
var schema_name = "";
var schema_json = "";
var issellable = false;
var istradable = false;
var isrevocable = false;
var isburnable = false;
var isactive = false;
var tags = "";
var category = "";
var image_file = "";
sdk.updateAssetSchemaWithImage(asset_schema_uid, schema_name, schema_json, issellable, istradable,
isrevocable, isburnable, isactive, tags, category, image_file).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UpdateAssetSchema

SDK Type: *Javascript/Node.js*

Description Updates an existing Asset Issuer.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*
- **schema_name** ~ *An Asset Schema's name.*
- **schema_json** ~ *A JSON Schema, which is required to create an Asset Schema https://json-schema.org/*
- **issellable** ~ *Enables minted Assets to be sold.*
- **istradable** ~ *Enables minted Assets to be traded between Users.*
- **isrevocable** ~ *Enables an Asset Issuer to remove a minted Asset from a User's inventory.*
- **isburnable** ~ *Enables an Asset Issuer to delete a minted Asset from a User's inventory.*
- **isactive** ~ *Enables Asset Schemas to be minted; if "Inactive", the developer is unable to mint new Assets.*
- **tags** ~ *List of tags for an associated Asset Schema; Max is 5.*
- **category** ~ *Category ID for a category.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
var schema_name = "";
var schema_json = "";
var issellable = false;
var istradable = false;
var isrevocable = false;
var isburnable = false;
var isactive = false;
var tags = "";
var category = "";
sdk.updateAssetSchema(asset_schema_uid, schema_name, schema_json, issellable, istradable,
isrevocable, isburnable, isactive, tags, category).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

RevokeInventoryAsset

SDK Type: *Javascript/Node.js*

Description Revokes an Asset within a user's inventory.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **asset_uid** ~ *An Asset within a User's inventory.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**
4. **MintAssetSchemaForUser**
5. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var asset_uid = "";
sdk.revokeInventoryAsset(asset_issuer_uid, asset_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

UseInventoryAsset

SDK Type: *Javascript/Node.js*

Description Uses an Asset within a User's inventory.

Required Inputs

- **asset_uid** ~ *An Asset within a User's inventory.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**
4. **MintAssetSchemaForUser**
5. **CreateGameSession**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_uid = "";
sdk.useInventoryAsset(asset_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

InviteUserFromAssetIssuer

SDK Type: *Javascript/Node.js*

Description Invites a User from an Asset Issuer's User System into the C0NTACT System's User System.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*
- **game_username** ~ *The username from an Asset Issuer's database to link to a User.*
- **game_uid** ~ *The Unique Identifier from an Asset Issuer's database to link a User.*
- **game_email** ~ *The email address of a User.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
var game_username = "";
var game_uid = "";
var game_email = "";
sdk.inviteUserFromAssetIssuer(asset_issuer_uid, game_username, game_uid, game_email).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

ListAssetIssuerAccounts

SDK Type: *Javascript/Node.js*

Description Lists all linked accounts for an Asset Issuer.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **LinkAssetIssuerAccountToUser**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
sdk.listAssetIssuerAccounts(asset_issuer_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

DisableAssetSchema

SDK Type: *Javascript/Node.js*

Description Disables an enabled Asset Schema.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
sdk.disableAssetSchema(asset_schema_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

EnableAssetSchema

SDK Type: *Javascript/Node.js*

Description Enables a disabled Asset Schema.

Required Inputs

- **asset_schema_uid** ~ *The UID of an Asset Schema, which is defined by the unique properties that make up a digital asset.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**
3. **CreateAssetSchema**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_schema_uid = "";
sdk.enableAssetSchema(asset_schema_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

DisableAssetIssuer

SDK Type: *Javascript/Node.js*

Description Disables a Enabled Asset Issuer.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
sdk.disableAssetIssuer(asset_issuer_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

EnableAssetIssuer

SDK Type: *Javascript/Node.js*

Description Enables a disabled Asset Issuer.

Required Inputs

- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var asset_issuer_uid = "";
sdk.enableAssetIssuer(asset_issuer_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

CreateOauth2Client

SDK Type: *Javascript/Node.js*

Description Allows a Developer to create a OAuth 2 Client.

Required Inputs

- **domain** ~ *The domain URL for the redirection system.*
- **description** ~ *Identifies the use of the OAuth2 Client.*
- **asset_issuer_uid** ~ *The UID of an Asset Issuer.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var domain = "";
var description = "";
var asset_issuer_uid = "";
sdk.createOauth2Client(domain, description, asset_issuer_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

GetUserDataOAuth

SDK Type: *Javascript/Node.js*

Description Gets a User's OAuth data.

Required Inputs

- **oauth_access_token** ~ *The token that is obtained via 0Auth2.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var oauth_access_token = "";
sdk.getUserDataOAuth(oauth_access_token).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

ListCategories

SDK Type: *Javascript/Node.js*

Description Lists all categories for an Asset Schema.

Required Inputs

Notes

> *Prerequisites are listed below, if any are required.*

1. **None**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);

sdk.listCategories().then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```

ListAssetIssuers

SDK Type: *Javascript/Node.js*

Description Lists all Asset Issuers for an Organization.

Required Inputs

- **organization_uid** ~ *The UID of an Organization.*

Notes

> *Prerequisites are listed below, if any are required.*

1. **CreateOrganization**
2. **CreateAssetIssuer**

```
var access = "00000000-0000-0000-0000-000000000000", secret = "00000000-0000-0000-0000-000000000000";
var contact = require('metropolis-node'), sdk = new contact(access, secret);
var organization_uid = "";
sdk.listAssetIssuers(organization_uid).then((data) => {
  console.log(data);
}).catch((err) => {
  console.log(err);
});
```