



TRACELINK UNIVERSITY

**Home**

**Resources**

**TraceLink University**

## XTT Link Actions Developer Kit



XTT Link Actions provide a generic, extensible framework for direct, automated communication between external systems (e.g. ERP, QMS, WMS) and TraceLink's Multienterprise Information Network Tower (MINT) solution.

The XTT Link Actions Developer Kit includes two tools, the Link Action Sandbox and the Link Action Test Tool, which allows local testing of Link Actions by verifying data based on a given configuration. The tool provides feedback in the form of success/fail messages during script execution with matching results added to the configuration files.

No roles are required to run the Link Action Sandbox, as this is a standalone application running outside of OPUS.

## Getting Started

You will require the latest version of the following software:

- **Git** (Necessary ONLY if cloning repositories by command line)
- **Node and NPM**

## Download Developer Kit

Start a terminal or command prompt session. Navigate to the folder that will hold

your test repository. Use the following command to create the folder:

Windows:

```
md link_action
```

MacOS:

```
mkdir link_action
```

### **Clone the individual repositories**

Navigate to the newly created folder:

```
cd link_action
```

Run the following commands to clone the repositories to your machine:

```
git clone https://github.com/tracelink/xtt-link-actions-sandbox
git clone https://github.com/tracelink/xtt-link-actions-test-tool
```

### **Install the Link Action Sandbox**

The Link Action Sandbox mimics the TraceLink Link Action framework that executes Link Actions, permitting calls to be made to the external system from a local machine. This sandbox is required to run the Link Action Test tool.

From the top level folder you just created, navigate to the Link Action Sandbox folder.

```
cd link-action-sandbox
```

Run the install command to download the required node packages.

```
npm install
```

Return to the top level directory to continue configuration.

```
cd ..
```

Navigate to the test tool folder and run the installation to gather the required

libraries for the tool:

```
cd link-action-test-tool  
npm install
```

This installation is only needed once. You are ready to run the tool as many times as needed.

## Configure the Link Action Developer Kit

### Configure Transaction Schema Files

Transaction Schema Files, called MaxFiles, are JSON files that supply an entry for every possible field, along with a sample value, of the Link Action transaction (e.g. Purchase Order, Invoice, ASN). MaxFiles are needed for validating Inbound Link Actions and are used to help generate schemas for testing.

Start with a JSON file containing the payload for a specific action such as creating a purchase order. To do this:

1. Copy the desired payload from the documentation for your application into an empty text file.
2. Edit your new file, placing a value in every possible field, including optional fields. The values need not be real data, but should match the type specified. Preserve all nesting or object hierarchies in your specified payload.
3. Save this new file in your test folder.

For example: the snippet below taken from a purchase order maxFile would be saved to **link-action-**

**sandbox/testData/Input/PurchaseOrder/Standard/Inbound/**

```
"billingAddress": {  
  "addr1": "baner",  
  "addr2": "stree 2",  
  "addressee": "BM Ltd.",  
  "addrPhone": "+1235345",  
  "addrText": "Tracelink \nBM Ltd.\nbaner\nstree 2\npune Maharashtra  
411002\nIndia",
```

```
"attention": "Tracelink",
"city": "pune",
"country": {
  "id": "IN",
  "refName": "India"
},
"override": false,
"state": "Maharashtra",
"zip": "411002"
}
```

Click [here](#) for a sample max file.

## Configure Link Action Test Tool

The Link Action Test Tool creates an integration with the customer's external system via the Link Action Sandbox to perform tests.

You must obtain credentials from your system of record for these tests to run properly. (see below if you are the administrator).

Navigate to the testData folder in the link action sandbox and copy the secrets.json file to the link-action-test-tool/testData folder

```
cd link-action-sandbox/testData
```

Windows:

```
copy secrets.json ../../link-action-test-tool/testData
```

MacOS:

```
cp secrets.json ../../link-action-test-tool/testData
```

Sample secrets.json file

```
{
  "API_KEYS": {
    "CONSUMER_KEY": "YOUR_CONSUMER_KEY",
    "CLIENT_SECRET": "YOUR_CLIENT_SECRET",
    "TOKEN_URL": "URL_FOR_YOUR_SERVICE",
    "CERTIFICATE_PRIVATE_KEY": "YOUR_GENERATED_PRIVATE_KEY",
    "CERTIFICATE_ID": "CERTIFICATE_ID_FROM_UPLOADED_PRIVATE_KEY",
  }
}
```

```
"ALGORITHM": "PS256",  
"SCOPE": "rest_webservices"  
}  
}
```

This file contains the credentials to facilitate communication with the external system for your test environment. You must supply the following items:

- **TOKEN\_URL** - URL to the service that generates the authentication token (e.g.  
`https://ab1234567.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token`, where ab1234567 is your company identifier)
- **CONSUMER\_KEY** - Value that identifies a specific user
- **CLIENT\_SECRET** - Value used to authenticate the CONSUMER\_KEY
- **CERTIFICATE\_PRIVATE\_KEY** - A cryptographic key that pairs with the uploaded public key. We create credentials in the next section.
- **CERTIFICATE\_ID** - The identifier for the certificate being used

To generate the access credentials and system configuration details required for the secrets.json file, an administrator for your external system may use the following guide:

- **Admin Setup for NetSuite**

Once the admin steps have been completed and the secrets.json file values have been added, save it to the link-action-test-tool/testData folder.

Configuration files

The **testData** subfolder within the **link-action-test-tool** folder contains configuration files required for the test to run. The configuration files are updated with success/fail messages during script execution.



The transform test tool uses Excel spreadsheets to contain the configurations for

each test. You must edit the configuration file for each test to reflect your current folder structure.

When you download the tool, there are sample configuration files that you may modify for your specific needs. The sample configurations allow you to test:

- Inbound Advanced Shipping Notice (ASN)
- Inbound Invoices
- Inbound purchase orders
- Outbound purchase orders
- Outbound sales orders

From the link-action-test-tool/testData folder, copy a sample file either in a graphical window or by running the copy command:

Windows:

```
copy NETSUITE_INVOICE_INBOUND_LINKACTION.xlsx  
My_Sample_INBOUND_INVOICE.xlsx
```

MacOS:

```
cp NETSUITE_INVOICE_INBOUND_LINKACTION.xlsx  
My_Sample_INBOUND_INVOICE.xlsx
```

Open the file for editing. On the **config** tab, update the folder location of your config.json file to reflect the location of your files. By default, it is in the link-action-sandbox/testData/Input folder in a subfolder that corresponds to your action. In the example below, the PurchaseOrder/Standard folder.



Before continuing, edit the config.json file you just referenced to contain the information relevant to your account.

Sample config.json

```
{
```

---

```
"url":
"https://ab1234567.suitetalk.api.netsuite.com/services/rest/record/v1"
,
  "netSuiteAccountId": "1234567",
  "erp": "Netsuite",
  "dateFormat": "YYYY-MM-DD",
  "timeFormat": "HH:MI:SS AM",
  "dateTimeFormat": "YYYY-MM-DD HH:MI:SS AM",
  "suiteQLURL":
"https://ab1234567.suitetalk.api.netsuite.com/services/rest/query/v1/suiteql"
}
```

Where ab1234567 is your account number. Use only the numeric portion of the AccountId for the **netSuiteAccountId** field.

On the **TestCase\_1** tab, update the locations for the following entries:

1. **MODIFY\_OBJECT** - The location of the configuration file containing the payloads needed to create records in the system of record.
2. **EXECUTE\_LINK\_ACTION\_INBOUND** - The location of the instruction file to execute the link action.
3. **CLEANUP** - The folder location the test tool uses for temporary files during testing. Files and folders in this location are deleted during script completion.



On the **data\_validation\_config** tab, update the values in each row of the SOURCE\_FILE column to match values for a record in your system of record.



The Link Action Developer Kit performs the following actions, which may be configured in the Excel files:

Action Name	Description
FETCH_OBJECT_INBOUND	Gets the data from ERP

MODIFY_CURSOR	Modifies the fetchedTillTime to a numeric representation of the current time
MODIFY_OBJECT	Creates a new record in the external system
EXECUTE_LINK_ACTION_INBOUND	Execute the link action
VERIFY_CURSOR	Valid only if MODIFY_CURSOR is used in an earlier step. Verifies the dates are current so that at least one record is returned.
VALIDATE_INBOUND	Compares the data from the external system against the output of the Link Action
CLEANUP	Deletes all files/folders under a given folder. It doesn't perform any cleanup activity on ERP side

## Running a test

Using a terminal window, navigate to the **link-action-test-tool** folder you created earlier.

Run the following command using a file that is provided for you:

```
node index.js --t testData/My_Sample_INBOUND_INVOICE.xlsx
```

The command line provides text feedback on the status of the progress and pass/fail results of each.



The script updates the test configuration file you created in the **testData** folder with pass or fail results. If the test fails, a reason is included.

The data\_validation\_config tab in the test configuration Excel file will produce similar output to the following on success:



The TestCase\_1 tab in the test configuration Excel file will produce similar output to the following on success:





The results of these tests should be used to confirm or rule out configuration issues as a root cause of communication problems with your link actions.

## Common Issues

Missing or incorrect data and unsuccessful token generation are common issues you may encounter. Below are some examples of error messages and a likely cause and resolution for each.

Sample error generated by using incorrect data:



In this situation, an incorrect invoice ID was sent. Fix this error by validating the information on the `data_validation_config` page of your test file against a valid record in your system.

Sample error generated by using incorrect client credentials:



This error was caused by using the incorrect `CONSUMER_KEY` value. Be sure to copy the correct values for your integration when creating it. `CONSUMER_KEY` and `CLIENT_SECRET` values are only displayed once.

Sample token generation error:



This error likely indicates a problem copying the `CERTIFICATE_PRIVATE_KEY` value into the `secrets.json` file. Be sure to remove any line breaks or carriage returns from the text of the private key when copying this value.

## Next Steps

Consult the [Custom XTT Link Actions Developer Guide](#) for more information on creating and working with Link Actions.