Automated Regression Testing with Worksoft Suite - Manhattan Scripts

Manhattan Scripts – How They Work

“Unique” might sum up the structure of the existing Manhattan automated regression testing scripts within Worksoft. There are several things that make the Manhattan automated scripts unique, see the glossary for details.

Glossary:

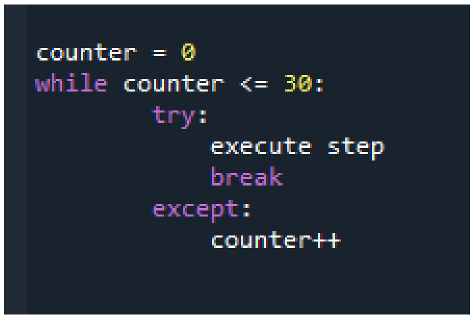
- Custom Timeout Loops
- Files
- GitHub
- XML Generation
- Naming Conventions

Back to Top

Custom Timeout Loops:

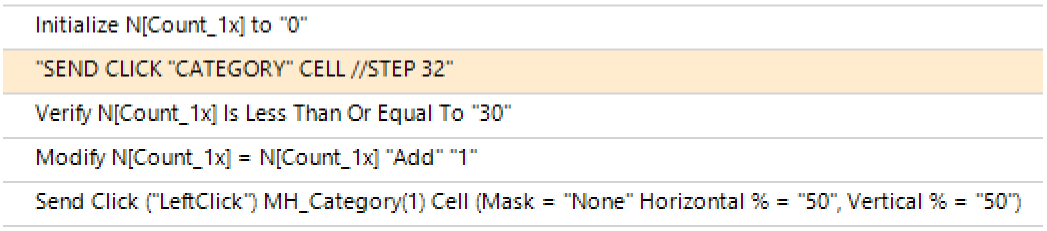
- There are a few "Action" settings that allow for a custom timeout to be applied to X step, but it is not on all of them. Certify does have the option for setting an explicit step timeout, but as the name suggests, this is applied to all steps in every process/subprocess in said script. With the Manhattan application having many different UIs/windows that load and run at varying speeds, an implicit timeout was needed.

With no implicit waits built into Certify, a custom loop was developed by Anthony Granato, which is basically just a while loop. In Python, it would look something like this:



```
counter = 0
while counter <= 30:
        try:
            execute step
            break
        except:
            counter++
```

In Certify, it looks like this:



```
Initialize N[Count_1x] to "0"
"SEND CLICK "CATEGORY" CELL //STEP 32"
Verify N[Count_1x] Is Less Than Or Equal To "30"
Modify N[Count_1x] = N[Count_1x] "Add" "1"
Send Click ("LeftClick") MH_Category(1) Cell (Mask = "None" Horizontal % = "50", Vertical % = "50")
```

If the Send Click step fails, it will jump up to the colored line and try again, iterating the counter each time (the top four lines being the same for every step that has this loop applied). The third line in the Certify example can, with this custom loop, be customized to compare against whatever number we want. 30 is about 10 seconds when Certify is having a good day, but again, we could set that to anything we want, depending on how long we want it to try executing X step. We do this for almost every step that requires something to be clicked on the screen. Without it, Certify would run faster than the application we test on, and steps would fail as a result. Note: there is also a Certify option for a step delay, but once again, with Manhattan having UI/windows that run at different speeds, it is not efficient to use such a setting as it is applied to all steps within a script. Although the custom timeout loops do add lots of extra rows to our "code," they allow for the fastest possible execution of X script.

Back to Top

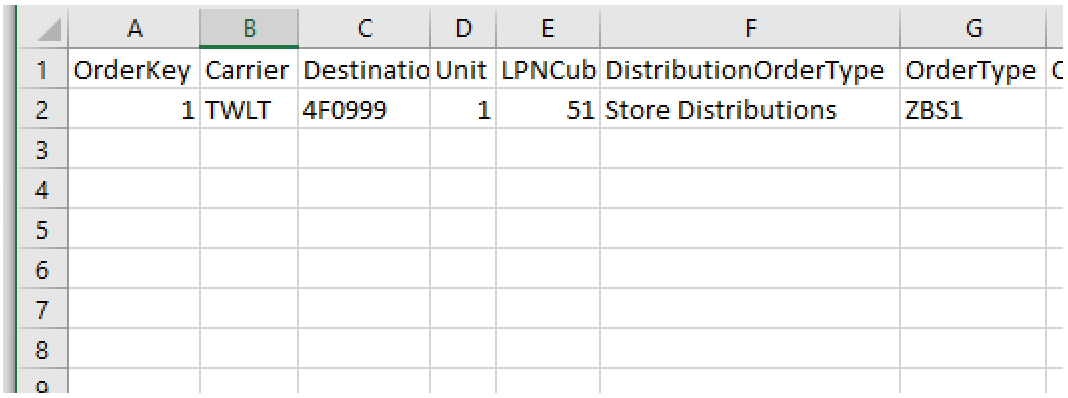
Files:

See the Recordsets section in the AutomationUsingWorksoftSuite file for more details and how the files discussed are populated and imported by scripts. For how these files are managed across multiple machines (including the remote machines we use for executing scripts), see the GitHub section in this file.

Input Files:

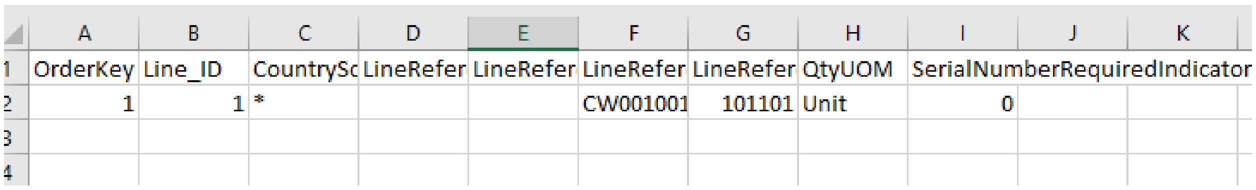
While efforts in automated test scripts are always focused on providing a working script with as little manual input as possible, there is always going to be some data/information that needs to come from the user. Most important of this user information is regarding ASN and DO creation. While input for DO creation may look the same for every script, removing the ability for the user to choose said variables actual aids in making the scripts less useful. See the XML Generation section below for how these input files are used, merely remember that almost all Manhattan scripts have at least one csv file from which data is collect from a user. For outbound scripts, there are two files, one for details on the Distribution Order (OrderDetails), and another that provides more details for DO creation, and ASN and/or LPN creation (LineDetails), as necessary. For inbound scripts, there is primarily one csv file used to provide specifics for ASN creation during script executions. Note, in the following examples: the first row is the variable name that values below said line will be applied to.

Example of outbound OrderDetails file:



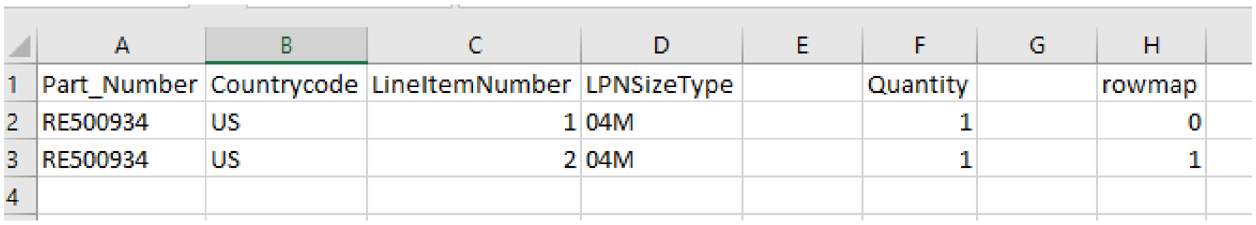
	A	B	C	D	E	F	G	
1	OrderKey	Carrier	Destinatio	Unit	LPNCub	DistributionOrderType	OrderType	C
2	1	TWLT	4F0999	1	51	Store Distributions	ZBS1	
3								
4								
5								
6								
7								
8								
9								

Example of outbound LineDetails file:



	A	B	C	D	E	F	G	H	I	J	K
1	OrderKey	Line_ID	CountrySc	LineRefer	LineRefer	LineRefer	LineRefer	QtyUOM	SerialNumberRequiredIndicator		
2	1	1	*			CW001001	101101	Unit	0		
3											
4											

Example of Inbound Inbound_P2P file:

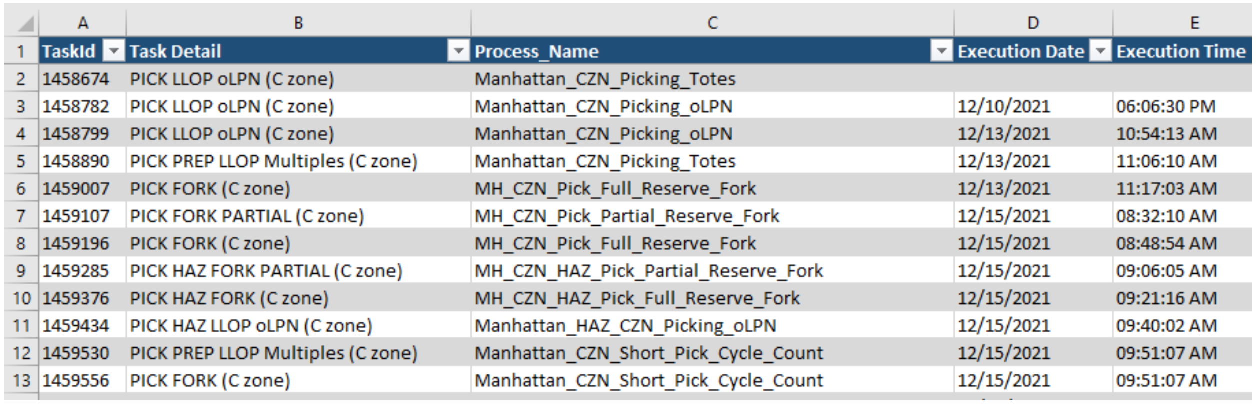


	A	B	C	D	E	F	G	H
1	Part_Number	Countrycode	LineItemNumber	LPNSizeType		Quantity		rowmap
2	RE500934	US	1	04M		1		0
3	RE500934	US	2	04M		1		1
4								

Output Files:

These files are far simpler than input files, in that, we are merely outputting data found/created in the scripts to excel files for later reference. In the case of the Manhattan scripts, we output every iLPN, oLPN, and TaskID into one file, along with data such as (if XML post fails) failed XML response messages.

Example:



	A	B	C	D	E
1	TaskId	Task Detail	Process_Name	Execution Date	Execution Time
2	1458674	PICK LLOP oLPN (C zone)	Manhattan_CZN_Picking_Totes		
3	1458782	PICK LLOP oLPN (C zone)	Manhattan_CZN_Picking_oLPN	12/10/2021	06:06:30 PM
4	1458799	PICK LLOP oLPN (C zone)	Manhattan_CZN_Picking_oLPN	12/13/2021	10:54:13 AM
5	1458890	PICK PREP LLOP Multiples (C zone)	Manhattan_CZN_Picking_Totes	12/13/2021	11:06:10 AM
6	1459007	PICK FORK (C zone)	MH_CZN_Pick_Full_Reserve_Fork	12/13/2021	11:17:03 AM
7	1459107	PICK FORK PARTIAL (C zone)	MH_CZN_Pick_Partial_Reserve_Fork	12/15/2021	08:32:10 AM
8	1459196	PICK FORK (C zone)	MH_CZN_Pick_Full_Reserve_Fork	12/15/2021	08:48:54 AM
9	1459285	PICK HAZ FORK PARTIAL (C zone)	MH_CZN_HAZ_Pick_Partial_Reserve_Fork	12/15/2021	09:06:05 AM
10	1459376	PICK HAZ FORK (C zone)	MH_CZN_HAZ_Pick_Full_Reserve_Fork	12/15/2021	09:21:16 AM
11	1459434	PICK HAZ LLOP oLPN (C zone)	Manhattan_HAZ_CZN_Picking_oLPN	12/15/2021	09:40:02 AM
12	1459530	PICK PREP LLOP Multiples (C zone)	Manhattan_CZN_Short_Pick_Cycle_Count	12/15/2021	09:51:07 AM
13	1459556	PICK FORK (C zone)	Manhattan_CZN_Short_Pick_Cycle_Count	12/15/2021	09:51:07 AM

Back to Top

GitHub

See the *"How to Manage Test Data Files using GitHub"* file within this same SharePoint folder for all the information on setting up and using GitHub (and how we use GitHub with Manhattan scripts).

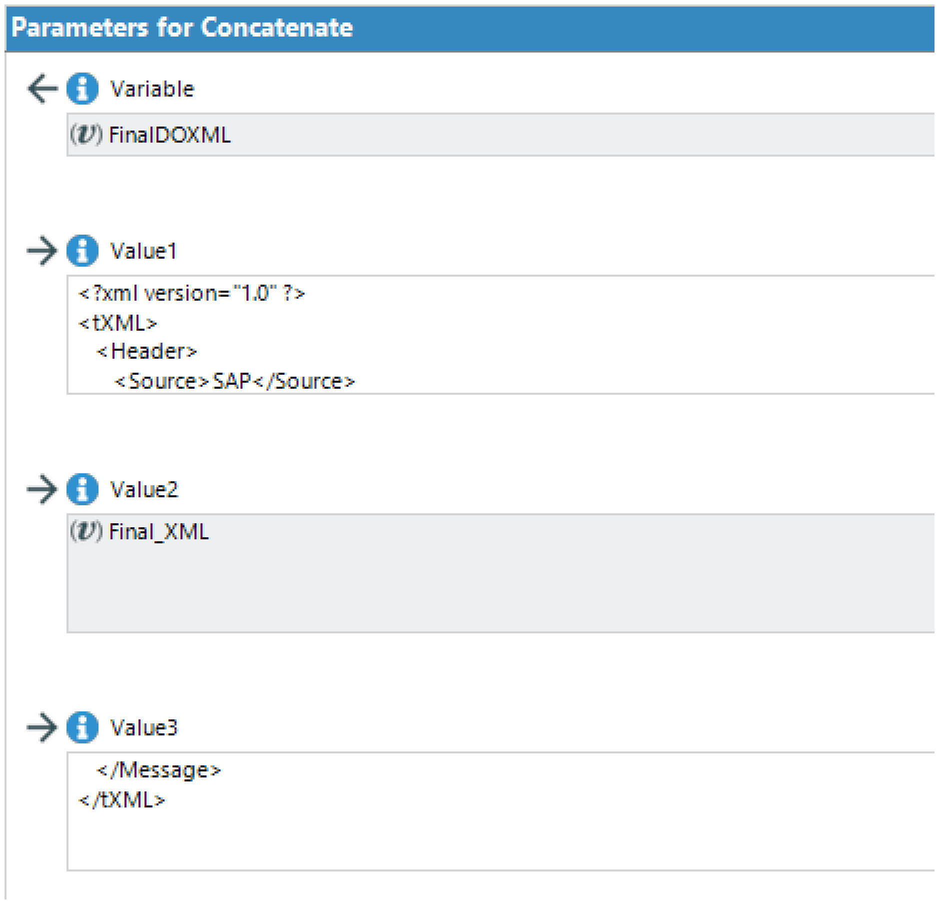
Alt link: How to Manage...using GitHub

Back to Top

XML Generation:

Understanding concatenation in coding terms will be very useful in understanding this section. XML is what makes the automated test scripts possible. For, XML posting within the Manhattan system allows for the creation of just about anything you want. In the case of the scripts, Distribution Orders, Advanced Ship Notices, and iLPNs. How and what XML is, and how it works is not the purpose of this section. Instead, I want to quickly walk through the basics of how the Manhattan scripts are generating their own XML. As mentioned in the Files section, some user input is needed to make XML, which is imported into a recordset, whose values are cross-referenced with Worksoft Certify variables and assigned to them. These variables are then concatenated with raw XML built into Certify steps within respective subprocess(es).

An example of concatenation in the wild:



Parameters for Concatenate

Variable
(V) FinalDOXML

Value1
```
<?xml version="1.0" ?>
<tXML>
  <Header>
    <Source>SAP</Source>
```

Value2
(V) Final_XML

Value3
```
  </Message>
</tXML>
```

Full XML:

Full outbound XML example:

```
<?xml version="1.0" ?>
<tXML>
    <Header>
```

Back to Top

```
        <Source>SAP</Source>
        <Action_Type>Update</Action_Type>
        <Message_Type>DistributionOrder</Message_Type>
        <Company_ID>01</Company_ID>
    </Header>
    <Message>
      <DistributionOrder>
           <DistributionOrderId>20Jun22AUTO747078</DistributionOrderId>
           <OrderType>ZBS1</OrderType>
           <OrderedDttm>11/09/2021 12:05:00 PM</OrderedDttm>
           <BusinessUnit>01</BusinessUnit>
           <MustReleaseByDttm>11/09/2021 12:00:00 AM</MustReleaseByDttm>
           <Priority>10</Priority>
           <ReferenceField1></ReferenceField1>
           <ReferenceField2>3Q0A</ReferenceField2>
           <ReferenceField3></ReferenceField3>
           <ResidentialDeliveryRequired>NEI</ResidentialDeliveryRequired>
           <OriginFacilityAliasId>CMP</OriginFacilityAliasId>
           <DestinationFacilityAliasId>4F0999</DestinationFacilityAliasId>
           <DestinationFacilityName>JOHN DEERE EQUIPAMENTOS DO B</DestinationFacilityName>
           <DestinationAddressLine1>ALAMEDA CAIAPOS 298</DestinationAddressLine1>
           <DestinationAddressLine2>TAMBORE, BARUERI</DestinationAddressLine2>
           <DestinationAddressLine3></DestinationAddressLine3>
           <DestinationCity>SAO PAULO</DestinationCity>
           <DestinationCounty></DestinationCounty>
           <DestinationCountry>BR</DestinationCountry>
           <DestinationStateOrProvince></DestinationStateOrProvince>
           <DestinationPostalCode>06460-110</DestinationPostalCode>
           <LpnCubingIndicator>51</LpnCubingIndicator>
           <BillToName>JOHN DEERE EQUIPAMENTOS DO B</BillToName>
           <BillToFacilityAliasId>4F0999</BillToFacilityAliasId>
           <DistributionOrderType>Store Distributions</DistributionOrderType>
           <DsgShipVia>TWLT</DsgShipVia>
           <LpnLabelType>SL</LpnLabelType>
           <ContentLabelType>OCL</ContentLabelType>
           <LineItem>
                <DoLineNbr>0001</DoLineNbr>
                <ItemName>KK17903</ItemName>
                <ReferenceField1></ReferenceField1>
                <ReferenceField2></ReferenceField2>
                <ReferenceField3>CW00100100</ReferenceField3>
                <ReferenceField4>101101</ReferenceField4>
                <SerialNbrRequiredIndicator>0</SerialNbrRequiredIndicator>
               <Quantity>
                     <OrderQty>37.0000</OrderQty>
                     <QtyUOM>Unit</QtyUOM>
               </Quantity>
               <InventoryAttributes>
```

Back to Top

```
                <CountryOfOrigin>*</CountryOfOrigin>
                <ItemAttribute1>*</ItemAttribute1>
                <ItemAttribute2>*</ItemAttribute2>
            </InventoryAttributes>
        </LineItem>
    </DistributionOrder>
  </Message>
</tXML>
```

Note: the yellow highlight is a value generated within the Manhattan scripts. However, virtually any of the tag values can be modifed, with many of the values coming from user input files.

Full inbound XML example (no LPNs being loaded in this example):

```
<?xml version="1.0" ?>
<tXML>
        <Header>
                <Source>Host</Source>
                <Action_Type>create</Action_Type>
                <Message_Type>ASN</Message_Type>
                <Company_ID>1</Company_ID>
        </Header>
        <Message>
                <ASN>
                        <ASNID>TEST235100089ASN</ASNID>
                        <ASNStatus>20</ASNStatus>
                        <BusinessUnit>1</BusinessUnit>
                        <OriginFacilityAliasID>CMP</OriginFacilityAliasID>
                        <DestinationFacilityAliasID>CMP</DestinationFacilityAliasID>
                        <ActualShippedDTTM>12/20/2019 00:00:00</ActualShippedDTTM>
                        <DeliveryStart>12/20/2019 00:00:00</DeliveryStart>
                        <DeliveryEnd>12/20/2019 00:00:00</DeliveryEnd>
                        <ExternalSysCreationDTTM>12/20/2019 00:00:00</ExternalSysCreationDTTM>
                        <OriginType>P</OriginType>
                        <DestinationType>W</DestinationType>
                        <IsCancelled>0</IsCancelled>
                        <ASNDetail>
                                <PurchaseOrderID>DemoPO001</PurchaseOrderID>
                                <PurchaseOrderLineItemID>1</PurchaseOrderLineItemID>
                                <ItemName>RE500934</ItemName>
                                <Quantity>
                                        <ShippedQty>1</ShippedQty>
                                        <QtyUOM>Unit</QtyUOM>
                                </Quantity>
                                </ASNDetail><ASNDetail>
                                <PurchaseOrderID>DemoPO001</PurchaseOrderID>
```

Back to Top

```
                                        <PurchaseOrderLineItemID>2</PurchaseOrderLineItemID>
                                        <ItemName>RE500934</ItemName>
                                        <Quantity>
                                                <ShippedQty>1</ShippedQty>
                                                <QtyUOM>Unit</QtyUOM>
                                        </Quantity>
                                </ASNDetail>
                        </ASN>
                </Message>
</tXML>
```

Inbound LPN example with LPNs loaded:

```
<?xml version="1.0" ?>
<tXML>
    <Header>
        <Source>Host</Source>
        <Action_Type>create</Action_Type>
        <Message_Type>ASN</Message_Type>
        <Company_ID>1</Company_ID>
    </Header>
    <Message>
        <ASN>
            <ASNID>A453884204N</ASNID>
            <ASNStatus>20</ASNStatus>
            <BusinessUnit>1</BusinessUnit>
            <OriginFacilityAliasID>CMP</OriginFacilityAliasID>
            <DestinationFacilityAliasID>CMP</DestinationFacilityAliasID>
            <ActualShippedDTTM>12/20/2019 00:00:00</ActualShippedDTTM>
            <DeliveryStart>12/20/2019 00:00:00</DeliveryStart>
            <DeliveryEnd>12/20/2019 00:00:00</DeliveryEnd>
            <ExternalSysCreationDTTM>12/20/2019 00:00:00</ExternalSysCreationDTTM>
            <OriginType>P</OriginType>
            <DestinationType>W</DestinationType>
            <IsCancelled>0</IsCancelled>
            <LPN>
                <LPNID>LPNT5567006</LPNID>
                <BusinessUnit>1</BusinessUnit>
                <LPNType>LPN</LPNType>
                <InboundOutboundIndicator>I</InboundOutboundIndicator>
                <DestinationFacilityAliasID>CMP</DestinationFacilityAliasID>
                <ProcessImmdNeeds>N</ProcessImmdNeeds>
                <PurchaseOrderID>AUTOMATED1</PurchaseOrderID>
                <PhysicalEntityCode>C</PhysicalEntityCode>
                <LPNDetail>
                    <ItemName>RE500934</ItemName>
```

Back to Top

```xml
                    <ItemSequenceNbr>1</ItemSequenceNbr>
                    <LPNDetailQuantity>
                        <Quantity>2</Quantity>
                        <QuantityUOM>Unit</QuantityUOM>
                        <ShippedAsnQuantity>2</ShippedAsnQuantity>
                    </LPNDetailQuantity>
                </LPNDetail>
                <LPNDetail>
                    <ItemName>RE199106</ItemName>
                    <ItemSequenceNbr>1</ItemSequenceNbr>
                    <LPNDetailQuantity>
                        <Quantity>2</Quantity>
                        <QuantityUOM>Unit</QuantityUOM>
                        <ShippedAsnQuantity>2</ShippedAsnQuantity>
                    </LPNDetailQuantity>
                </LPNDetail>
            </LPN>
        </ASN>
    </Message>
</tXML>
```

Back to Top

Naming Conventions

Naming is something that is very simple, but worth mentioning. MH denotes Manhattan when used. We put MH in front of all variables, objects, files, and processes so we know which are ours. At some point, something, or someone (could be one of us) will mess up and accidentally put one of the aforementioned items into a folder it does not belong in. Naturally, we want to keep things as tidy as possible, so any unnecessary items (i.e., those that do not belong in our Manhattan folders) should be removed to keep everything neat. While this may seem tedious, it is highly recommended so that nothing gets lost in any of the Certify UIs - either MH items in the wrong folder, or someone else's items in one of the MH folders, etc...

Multi User Login

Functionality to allow different users to login across end-to-end script executions at particular points as been added to inbound and outbound scripts. The reason: to test user permissions for various security groups (e.g., office worker, floor worker, etc.). An example to test different security groups effectively would be to switch users within our scripts to match that of the function being complete. Meaning, user in office security group would wave orders, and user in floor worker security group would complete the task. Another user from the shipping office would create appropriate shipment and close it, etc.

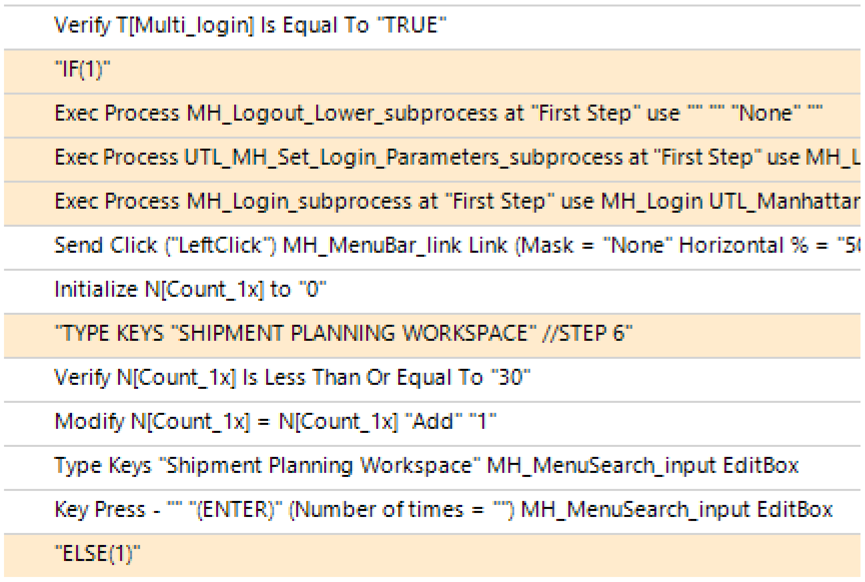
The first part relates to the Multi_login variable. In our Login.csv file, this variable can be set to a TRUE or FALSE value on column E for however many rows we have – this essentially indicates that we need to login with a new user.

Next, the User_function variable, which can be adjusted in the Login.csv file on column D. There are several user functions available to choose. Note, these are case sensetive:

1. Setup
2. Waving
3. Picking
4. Shipping
5. Loading
6. Shipment Closing

In process, this appears as:

Back to Top



```
Verify T[Multi_login] Is Equal To "TRUE"
"IF(1)"
Exec Process MH_Logout_Lower_subprocess at "First Step" use "" "" "None" ""
Exec Process UTL_MH_Set_Login_Parameters_subprocess at "First Step" use MH_L
Exec Process MH_Login_subprocess at "First Step" use MH_Login UTL_Manhattar
Send Click ("LeftClick") MH_MenuBar_link Link (Mask = "None" Horizontal % = "5(
Initialize N[Count_1x] to "0"
"TYPE KEYS "SHIPMENT PLANNING WORKSPACE" //STEP 6"
Verify N[Count_1x] Is Less Than Or Equal To "30"
Modify N[Count_1x] = N[Count_1x] "Add" "1"
Type Keys "Shipment Planning Workspace" MH_MenuSearch_input EditBox
Key Press - "" "(ENTER)" (Number of times = "") MH_MenuSearch_input EditBox
"ELSE(1)"
```

Note the Multi_login variable being checked first. If returned as FALSE, the top verify step will jump the execution down to the ELSE(1) statement, and the IF(1) step if the value is TRUE. Simply, an if:else statement. The Logout_Lower_subprocess will simply logout of the application without any of the data pumping that happens in the regular Logout_subprocess (seen at the end of every script). Next, login parameters will be set with a filter... in this case, it will set the username and environment for the row where User_function equals "Shipment Closing" (or another of the six options above depending where in the script we are checking Multi_login. E.g., at the beginning of MH_Pick_&_Anchor_oLPN(s), the filter for setting login parameters is User_function equals "Picking." So on and so on. Note within the Set_Login_Parameters_subprocess, the Multi_login variable will be set to whatever is on the selected row.... if we are setting user for picking process and Multi_login for that row is FALSE, when we get to shipping, the Multi_login verification will fail.

Example of what Login.csv would look like during multi user testing:

Environment	Username	Purchase_Order	User_function	Multi_login
devl	ADYMH2T	AUTOMATED1	Setup	TRUE
devl	H93IADW	AUTOMATED1	Waving	TRUE
devl	ADYMH2T	AUTOMATED1	Picking	TRUE
devl	EBU312Q	AUTOMATED1	Shipping	TRUE
devl	H93IADW	AUTOMATED1	Loading	TRUE
devl	ADYMH2T	AUTOMATED1	Shipment Closing	TRUE

Back to Top