

Transaction Designer

Version 4.3.0.9

User's Guide
Revision 4.9

CLEO

Copyright ©2011 Cleo Communications - Cleo Communications reserves the right to, without notice, modify or revise all or part of this document and/or change product features or specifications and shall not be responsible for any loss, cost or damage, including consequential damage, caused by reliance on these materials. This document may not be reproduced, stored in a retrieval system or transmitted, in whole or in part, in any form or by any means (electronic, mechanical, photocopied or otherwise) without the prior written permission of Cleo Communications.

GOVERNMENT RESTRICTED RIGHTS

Use, duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Use, reproduction or disclosure is subject to 52.227-19 (a) through (d) and restrictions set forth in the accompanying end-user agreement.

GOVERNMENT LIMITED RIGHTS

Limited rights shall be effective indefinitely and are not subject to expiration as set forth in paragraph (3) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Cleo Communications has made every effort to accurately acknowledge all trademarks that appear in this document. Cleo Communications, however, cannot attest to the accuracy of this information.

Cleo is a registered trademark of Cleo Communications. All other brand names are trademarks or registered trademarks of their respective owners.

Table of Contents

Introduction	1
Component Diagram	6
Key Features	6
Using the Transaction Designer	8
Run the Transaction Designer	8
Create a New Transaction Set	8
Open an Existing Transaction Set.....	9
Open an Existing Transaction Set from Previous Version of TD	9
Import Transactions from another Transaction Set.....	10
Import Screens Captured by Transaction Processor	11
Delete an Existing Transaction Set.....	11
Access the Screen Recorder.....	12
Configuring for Alternate Screen Sizes	13
Record Host Screens	16
Start and Stop Recording.....	17
Manipulate Recorded Screen Clips.....	18
Create a Transaction.....	18
Create New Transaction	20
Add Blank or Any Screen.....	21
Change Screen or Transaction Name.....	21
Create Screen Definitions.....	22
List Screens Using a Base Screen.....	24
Define Fields.....	25
Define Fields Received from the Host.....	25
Define Fields Sent to the Host	25
Highlight Input Fields.....	26
Define Table Driven Input Field Values	27
Define Actions	28
Duplicate Screen Definitions	34
Publish the Transaction Set.....	35
Transfer the Published Transaction Set.....	35
Save Transaction Set under Different Name.....	37
Run the Validator	37
Show Logs	43
Wait Time.....	43
Specify Macros.....	43
Menu Bar Descriptions	45
File Menu	45
Help Menu.....	46
Edit Menu	46
Appendix A: Troubleshooting	47
General Errors	47
Recorder Errors	55
Restrictions.....	56
Appendix B: Sample Transaction Map	57
Transaction Set: sample	57
Transaction: login	57
Transaction: park.....	58
Transaction: logout.....	58
Transaction: recovery.....	58
Transaction: search.....	58
Transaction: search_park.....	59
Transaction: nextScreen.....	60

Transaction: nextScreen_park.....	61
Transaction: getAllScreens.....	61
Transaction: getAllScreens_park.....	62
Appendix C: Conditional Transactions.....	64
Creating Branches.....	64
Creating Rules.....	67
Appendix D: Publishing VoiceXML.....	74
VoiceXML Generator.....	74
Functions Provided.....	75
Glossary.....	78
Index.....	80

Introduction

Programs that access mainframe data with host applications generally use one of IBM's Application Program Interfaces (APIs) such as the High Level Language API (HLLAPI). Cleo's Transaction Development Kit is a set of tools that provides a simplified mechanism for defining and processing custom host "transactions". These transactions contain all the information necessary to interact with a set of host display screens. Thus, once transactions have been created, the developer can send and extract mainframe data using fewer API calls than traditionally required. This simplifies the development process that in turn leads to reduced development time and higher reliability.

The Cleo development kit is composed of:

- *A Transaction Designer (TD)* – that records screens and creates transactions.
- *A Transaction Processor (TP)* – that processes the transactions in real time and provides the basic API functions.
- *An Administration GUI* - that provides a Configuration tool and an Administration tool for the Transaction Processor.

The documentation set that supports the development kit consists of the following:

Quick Start Guide for Transaction Designer

This guide lists prerequisites and provides an installation procedure for the Transaction Designer.

Cleo Transaction Designer User's Guide

This guide explains how to record screen clips and create transactions.

Quick Start Guide for Transaction Processor

This guide lists prerequisites and provides an installation procedure for the Transaction Processor.

Administration Guide

This document describes the web-based administration utility for the Transaction Processor product.

Programmer's Guide

This guide provides information about developing applications using the Transaction Processor.

This manual, *The Transaction Designer User's Guide*, contains the detailed instructions for using the Transaction Designer.

The intended audience for this document is presumed to be familiar with application development on a Windows platform, as well as having some knowledge of using a host emulator for accessing applications on a mainframe host.

Component Diagram

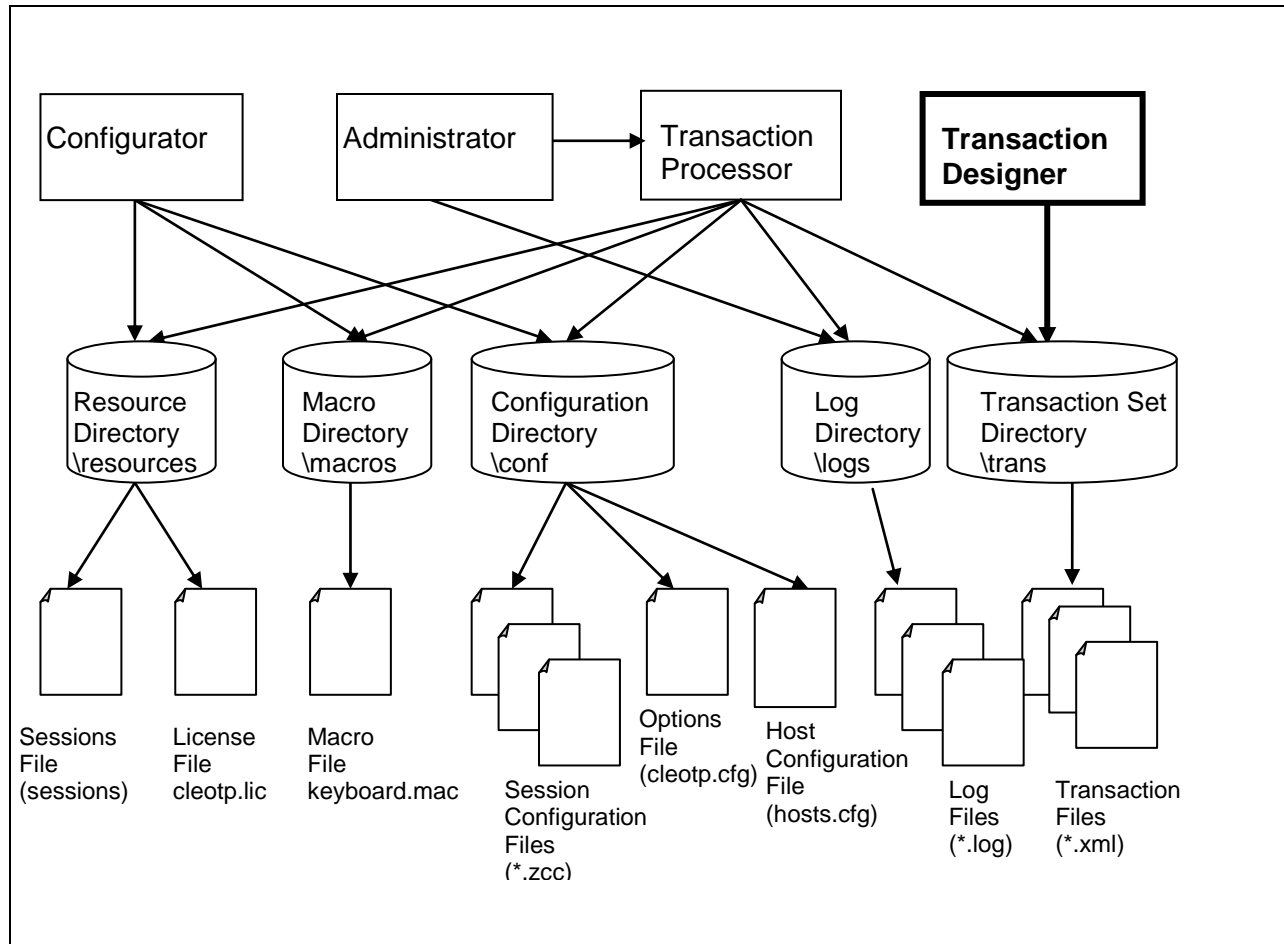


Figure 1: Component Diagram

Key Features

The Transaction Designer provides:

- A way to create recording clips consisting of a sequence of host screens that are recorded as the host application is manually navigated. The capability to start and stop automatic recording of host screens is provided. Each recording clip includes:
 - a representation of the host screen contents
 - keyboard input
 - the AID key used to navigate to the next screen
- A display of recorded screens with input fields highlighted
- A means to create/modify host transactions for each transaction set, such as “Login,” “Logout,” “Recovery,” “Park,” or a user-specified name.

- A means to save screen definitions and host transactions to files in a specific XML format, for use by the Cleo 3270 Plug-in or Transaction Processor.
- Options for running host emulators required to interact with the mainframe locally, or on the remote IR system.
- A Validator providing a way to test transactions using a live host connection.

Using the Transaction Designer

This section describes how to use the Transaction Designer. It includes how to:

- Record Screen Images
- Create Transaction Sets
- Publish Transaction Sets
- Validate “published” Transaction Sets

Run the Transaction Designer

This is the initial screen of the Cleo Transaction Designer (TD).

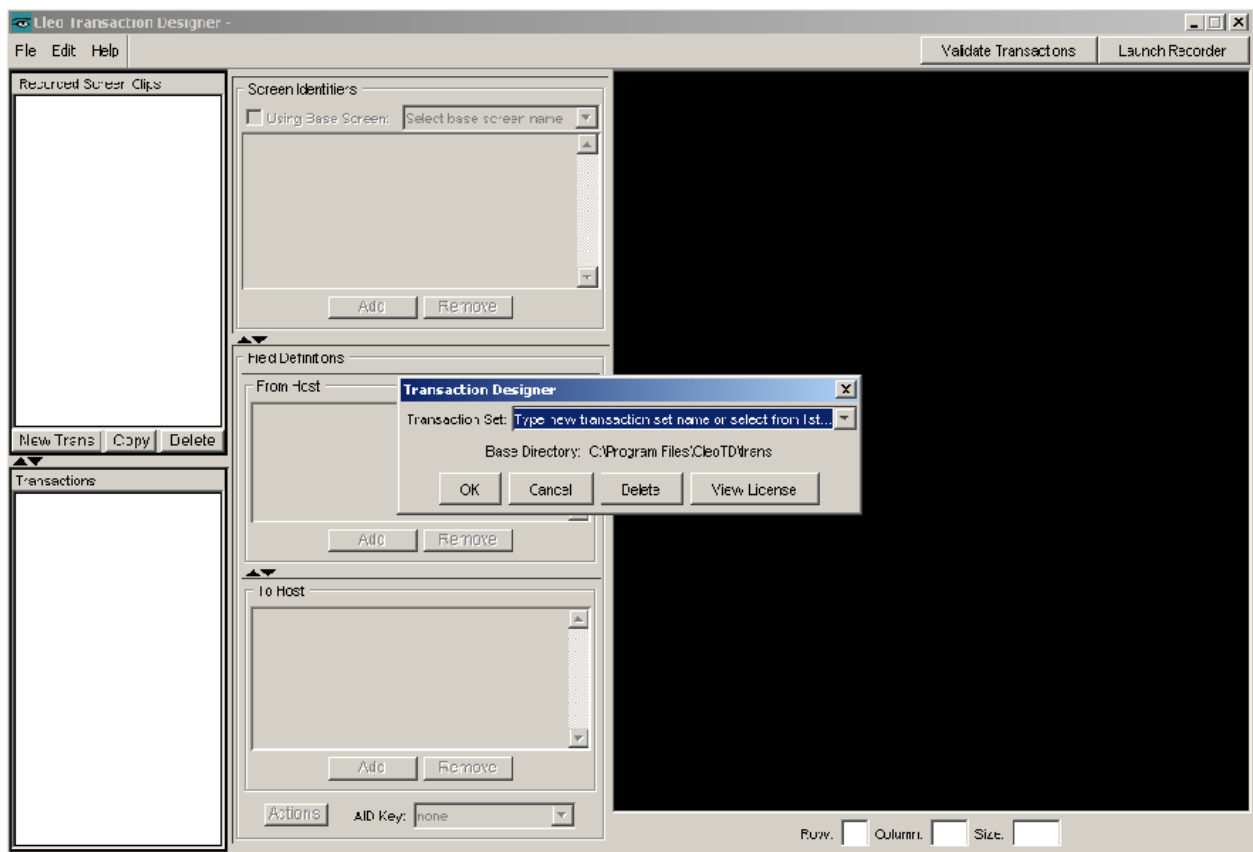


Figure 2: Initial Screen

From this screen, you may create a new transaction set, or specify an existing transaction set from the pull-down list.

Create a New Transaction Set

To create a new transaction set, perform the following:

1. Highlight <**Type new transaction set name...**> in the pull-down box.
2. Type a name for the transaction set. The **Path** displayed is the default. It identifies where the transaction set is stored. **Recommendation:** Choose a name that is similar, but not identical to the name of the IR application. This is useful in cross-referencing the IR application to the corresponding TD transaction set, but it is not necessary.
3. Click **OK**.

NOTE: If the CANCEL button is selected, the Transaction Designer will close down.

4. If the Transaction Set is to be created using newly recorded screens, then go to Access the Screen Recorder Section on Page 11. Otherwise, import an existing Transaction Set as the basis for the new Transaction Set using one of the following operations.

Open an Existing Transaction Set

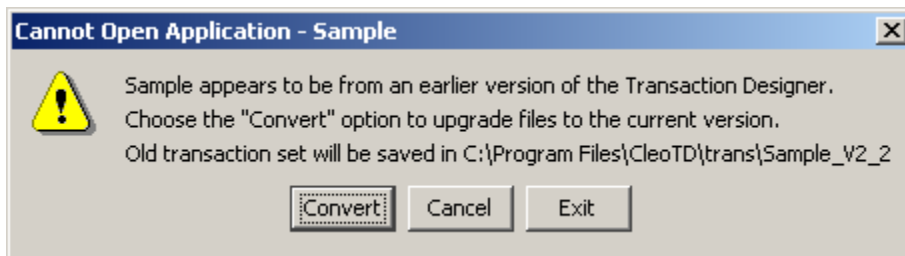
Open an already created transaction set by selecting (i.e., highlighting) the appropriate transaction set name displayed in the pull-down list. Then click **OK**:



Figure 3: Transaction Set Name Selection

Open an Existing Transaction Set from Previous Version of TD

Upon opening a transaction set from a previous version of the Transaction Designer, you will see the following message:



Simply click on **Convert** and the transaction set will be automatically converted for you. The old version of your transaction set will be stored in the *trans* directory in a folder with the extension *_V2_2* appended to the name of the transaction set.

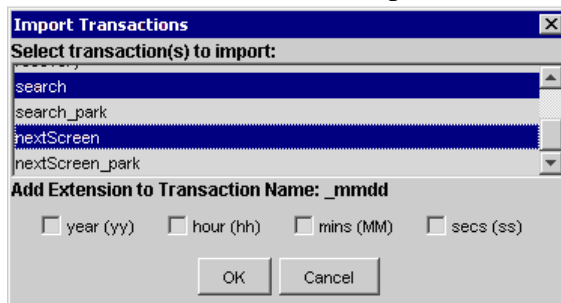
Import Transactions from another Transaction Set

To import transactions (along with their associated recorded screens) from another transaction set into the current transaction set:

1. Select File | Import | Transactions from the menu bar
2. Select the transaction set from which you wish to import transactions from.



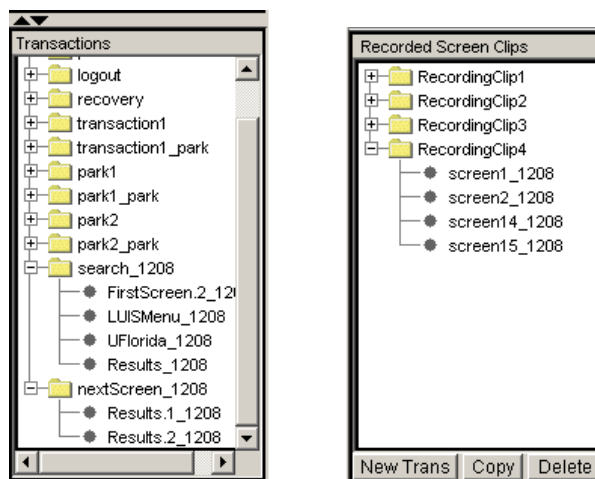
3. Click OK.
4. Select the transaction(s) to import.



5. The required extension is _mmdd. (eg. “search_1208” if the import occurs on December 8th). You may optionally select additional extensions to add to the transaction name.

Note: The extension is necessary to prevent two transactions from having the same name once the selected transaction is imported.

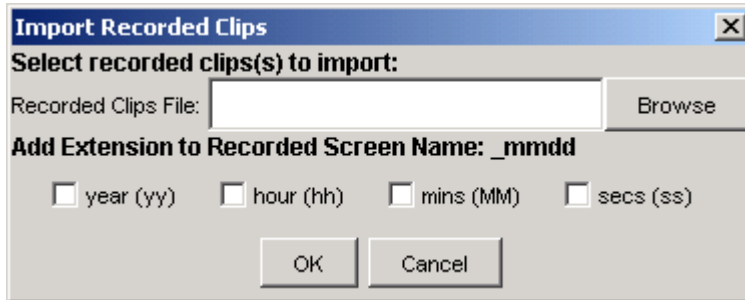
6. Once the import is complete, the imported transactions will appear in the Transactions panel and the associated recorded screens will show up in a new recording clip in the Recorded Screen Clips panel.



Import Screens Captured by Transaction Processor

Unrecognized screens received from the host by the Transaction Processor (TP) are saved in files having the “htp” extension in the “scr_dump” folder where the TP was installed. To view these screens, these files must be imported into the current transaction set as a Recording Clip, as follows:

1. Select File | Import | Recorded Clips from the menu bar.
2. Specify the location of the *.htp file to be imported, using the following dialog:



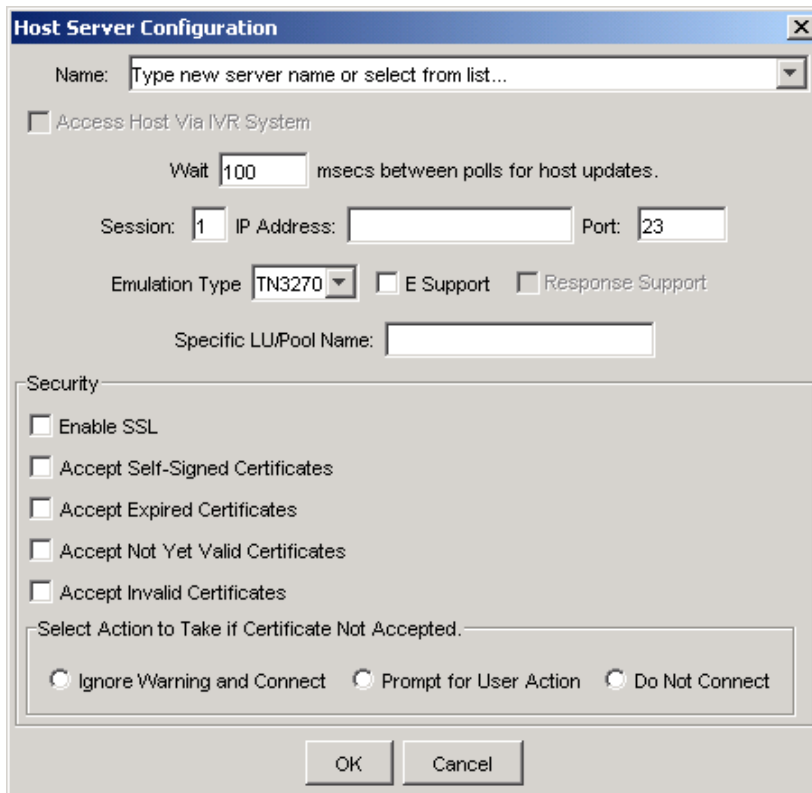
3. Use the Browse button to specify the complete path of the file and click OK.
4. A new Recording Clip is added to the Recorded Screen Clips panel. Click on the screen name to display the contents in the right panel.

Delete an Existing Transaction Set

To delete a transaction set and its associated files, highlight the transaction set name and click **Delete**.

Access the Screen Recorder

The Screen Recorder functions like a host emulator to provide access to a host application. To run the Screen Recorder, click on **Launch Recorder**. The following host server configuration dialog is displayed:



The dialog box is titled "Host Server Configuration" and contains the following fields and options:

- Name:** A text box with a drop-down arrow, containing the text "Type new server name or select from list..."
- Access Host Via IVR System**
- Wait:** A text box containing "100" followed by "msecs between polls for host updates."
- Session:** A text box containing "1"
- IP Address:** A text box
- Port:** A text box containing "23"
- Emulation Type:** A drop-down menu showing "TN3270"
- E Support**
- Response Support**
- Specific LU/Pool Name:** A text box
- Security:** A section containing:
 - Enable SSL**
 - Accept Self-Signed Certificates**
 - Accept Expired Certificates**
 - Accept Not Yet Valid Certificates**
 - Accept Invalid Certificates**
 - Select Action to Take if Certificate Not Accepted:**
 - Ignore Warning and Connect**
 - Prompt for User Action**
 - Do Not Connect**
- OK** and **Cancel** buttons at the bottom.

Figure 4: Host Server Configuration Dialog

Select an existing name from the drop-down box, or type a new name. If a new host server name is typed, the following information must be entered:

- **IP Address:** TCP/IP address of tserver or host connection
- **Port:** TCP/IP port of tserver or host connection
- **Session:** session number in an emulator (for connection). The session number range is 1-10.
- **Emulation Type:** TN3270 or TN5250
- **Wait:** amount of time screen recorder tool waits for a response from the host after sending an AID key. The default is 100 milliseconds
- **E Support:** Indicates that the host supports extended capabilities for TN3270 or TN5250
- **Specific LU/Pool Name:** Name of pool or LU configured on tserver/host (requires extended capabilities)
- **Response Support:** Response support is required by the tserver/host (requires extended capabilities)

- **Enable SSL:** When checked, SSL (Secure Sockets Layer) encryption is enabled and both TN3270 and TN5250 terminal emulation sessions are protected from eavesdropping, tampering, or message forgery. Secure Telnet sessions using SSL typically use TCP Port 992 instead of 23, but may be re-configured to any available TCP Port desired. The host mainframe computer you are connecting to must also be configured for SSL, or a connection will not be made.

The following are ignored if the Enable SSL box is not checked:

- **Accept Self-Signed:** Check to accept a self-signed server certificate
- **Accept Expired:** Check to accept a server certificate that has expired
- **Accept Not Yet Valid:** Check to accept a server certificate that has a starting date in the future
- **Accept Invalid:** Check to accept invalid server certificates for any reason other than the date or signature. The server certificate check will be ignored.
- **Select Action to take if certificate not Accepted:**
 - Select **Ignore Warning and Connect** if connection should be made to the host regardless of certificate acceptance.
 - Select **Prompt for User Action** to view the certificate and choose whether to accept it or not.
 - Select **Do Not Connect** if certificate is not accepted. No connection is made to the host and the screen will be blank.

Configuring for Alternate Screen Sizes

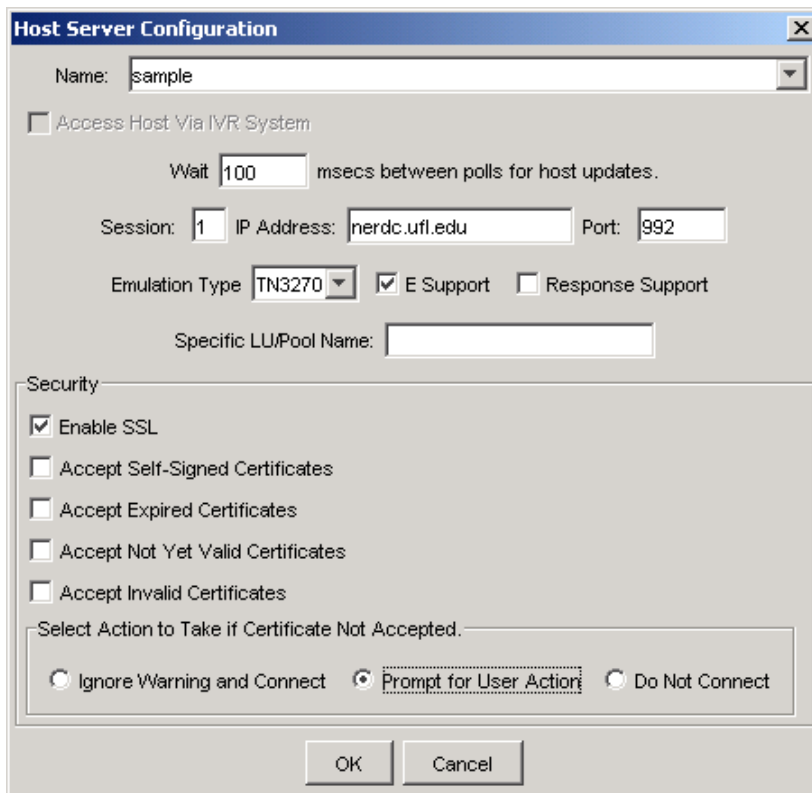
If a terminal screen size greater than 24 rows by 80 columns (mod2) is to be used, the host configuration file (sess<n>.zcc) must be edited to change the 3270ScreenSize parameter from 2 to one of the following:

- 3** for mod3 (32 rows by 80 columns)
- 4** for mod4 (43 rows by 80 columns)
- 5** for mod5 (27 rows by 132 columns)

For example, to use session id 4 to access a mod4 screen, the following steps are taken:

1. Edit sess4.zcc in the folder where the TD was installed (default is C:\Program Files\CleoTD) and change the line “3270ScreenSize=2” to “3270ScreenSize=4”.
2. Launch the recorder and specify session 4 in the Host Server Configuration dialog.
3. Enter the other host connection parameters as described above.

Figure 5 is an example of a completed dialog used to connect to a 3270 session on a public tn3270 site providing a library card catalog system. The “sample” transaction set was created using this host connection.



The image shows a 'Host Server Configuration' dialog box with the following fields and options:

- Name: sample
- Access Host Via IVR System
- Wait 100 msec between polls for host updates.
- Session: 1 IP Address: nerdc.ufl.edu Port: 992
- Emulation Type: TN3270 E Support Response Support
- Specific LU/Pool Name: [empty]
- Security section:
 - Enable SSL
 - Accept Self-Signed Certificates
 - Accept Expired Certificates
 - Accept Not Yet Valid Certificates
 - Accept Invalid Certificates
 - Select Action to Take if Certificate Not Accepted:
 - Ignore Warning and Connect
 - Prompt for User Action
 - Do Not Connect

Buttons: OK, Cancel

Figure 5: Sample Host Server Configuration

After clicking **OK**, if the connect was successful, the host screen will be displayed and the status and transaction set lines will be updated. If not, an error is displayed indicating a possible cause and resolution of the problem. If the screen is unexpectedly blank, retry by selecting Connect under the **Session** menu item and entering the correct IP address, port, or LU name. It is important that the Host Server Configuration parameters agree with the particular mainframe to which you wish to connect. See Recorder Errors

Example screen:

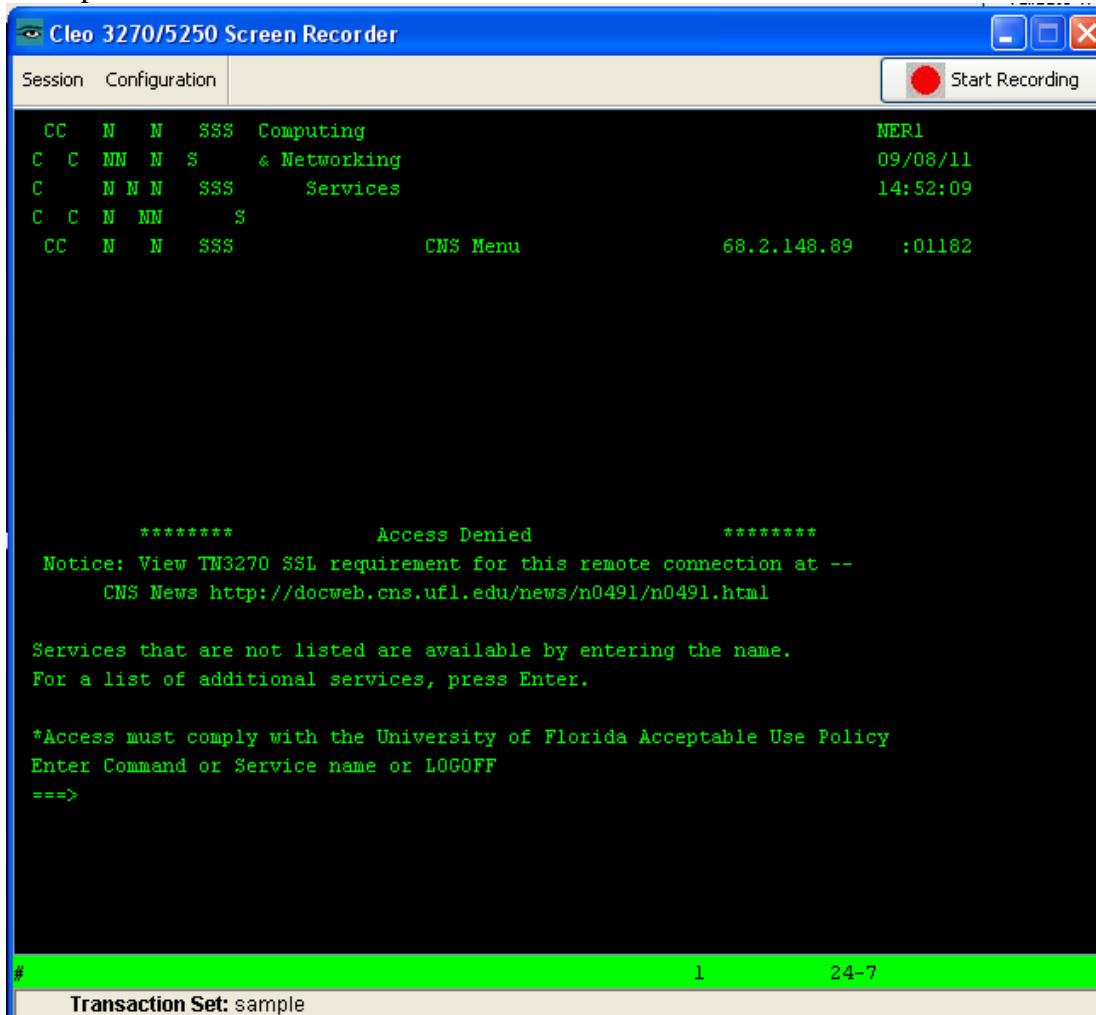


Figure 6: Host Screen Example

The following functions are provided under the **Session** menu to navigate the host application:

- **Redraw:** There may be instances where the utility does not reflect the current host screen (i.e. is out of sync). Redraw will refresh the presentation space so that it contains the correct screen information.

NOTE: This is an emulator function. No information is sent or received from the host.

- **Connect:** Allows the user to retry or change the host connection. The dialog box in Figure 4 is displayed.

The following functions are provided under the **Configuration** menu item:

- **Keyboard Map:** Allows you to view the current 3270 keyboard map, shown below, while using the Screen Recorder Tool. Click **X** to close the dialog box.

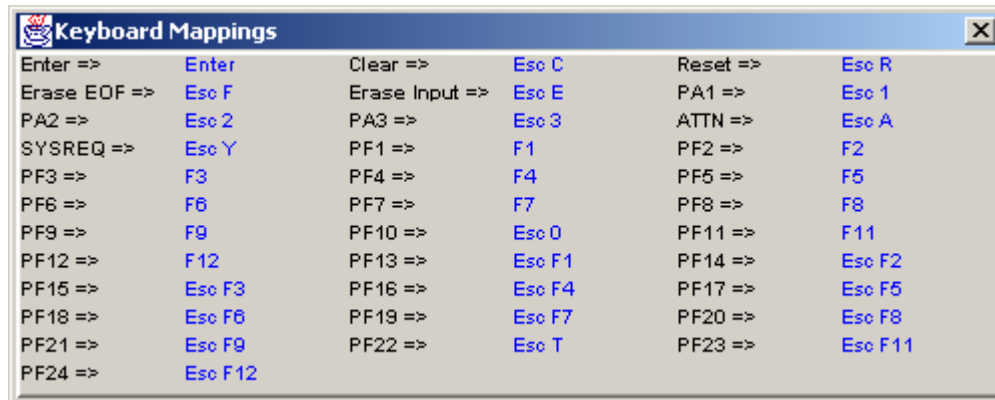


Figure 7: Keyboard Mappings

- **Default Settings:** Provides settings used when creating a 3270 Server Configuration.



Figure 8: Default Settings

Port: TCP/IP port number of tserver or host connection.

Wait: amount of time screen recorder waits for a response from the host after sending an AID key. The default is 100 milliseconds.

Record Host Screens

After connecting to the host session, you will need to create a recording of all host screen images that are to be used in the transaction set. The Host Screen Recorder allows you to navigate to the desired beginning of a host transaction, and then start the recording of host screen images while continuing to navigate through the host application. Figure 9 illustrates the screen when in recording mode, indicating that the host screen image displayed has been recorded under the name “screen1”. Screen names default to “screen” appended with its number, assigned in consecutive order starting with 1.

After making recordings, leave the Screen Recorder window open to refer back to the host application during the transaction definition process. Or, close the window by clicking the **X**

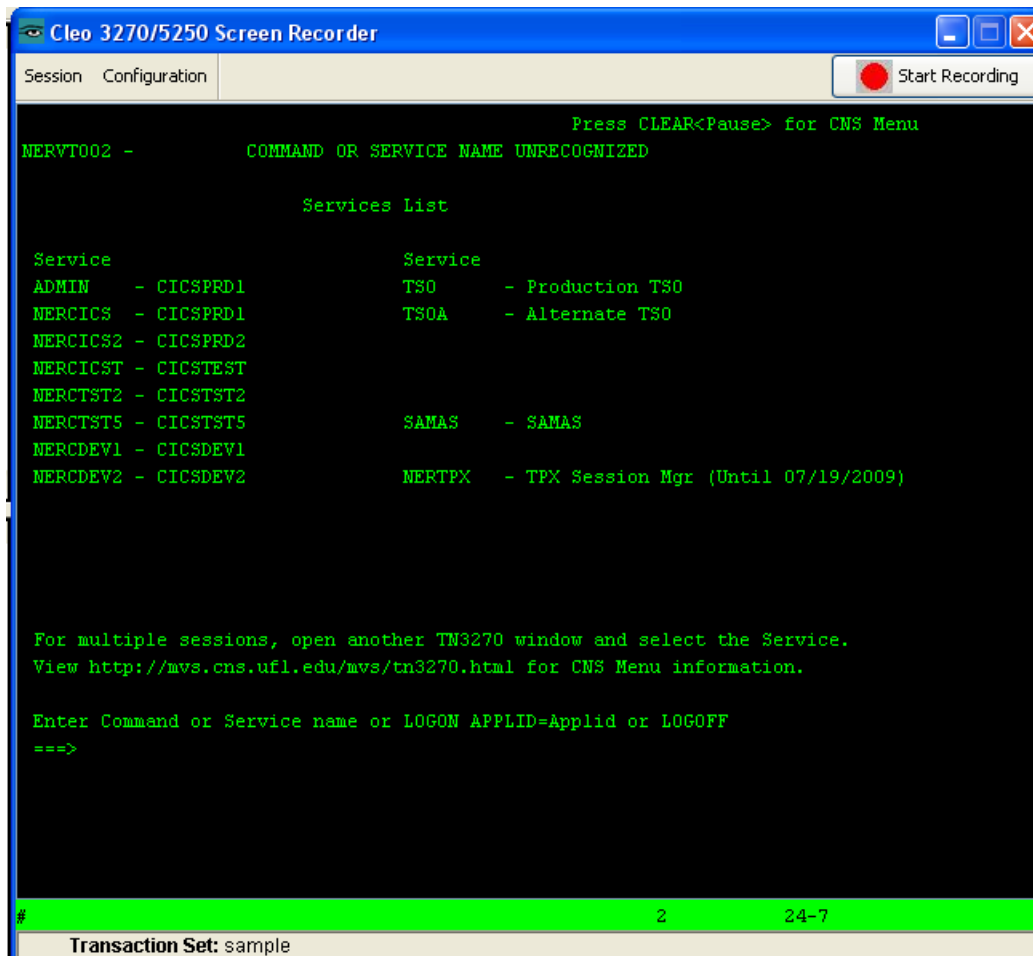


Figure 9: Sample Recorded Host Screen

Start and Stop Recording

The toggle button in the upper right hand corner of the toolbar controls whether recording is on or off. To start recording the screen contents, keyboard input, and AID key entries:

- click **Start Recording**. (The label changes to **Stop Recording**. The Transaction Set bar will display a red RECORDING indicator and also indicate what screen number is being recorded.)

To stop recording:

- click **Stop Recording**.

NOTE: The *STOP RECORDING* button is enabled only when:

1. The **Start Recording** button has been clicked
2. The keyboard is enabled and unlocked

Manipulate Recorded Screen Clips

The Recorded Screen Clips contain a collection of host screens (“recording clips”) that are recorded during a single recording session. A recording clip ends when recording stops (by clicking on **Stop Recording**).

A new recording clip is started by clicking **Start Recording**. Each new clip is assigned the name “RecordingClip” appended with its number, assigned consecutively starting with 1.

As each host screen is recorded, it is saved, and its name is added to the display of Recorded Screen Clips in the Transaction Designer window. The recorded screen can then be added to a host transaction, as described in the next section, “Create a Transaction.”

NOTES:

- *The last screen in a recording clip will not contain an aid key since one was never entered.*
- *Exiting the Screen Recorder Tool implies selection of the **Stop Recording** button.*

In the event that a recording produces screens that will not be used in any host transaction, the recording clip that contains those screens can be deleted.

To delete a recording clip:

1. Select the recording clip that is to be deleted.
2. Select **Edit| Delete Recording Clip** from the menu.
3. The following prompt will appear if screens within the recording clip are being used:

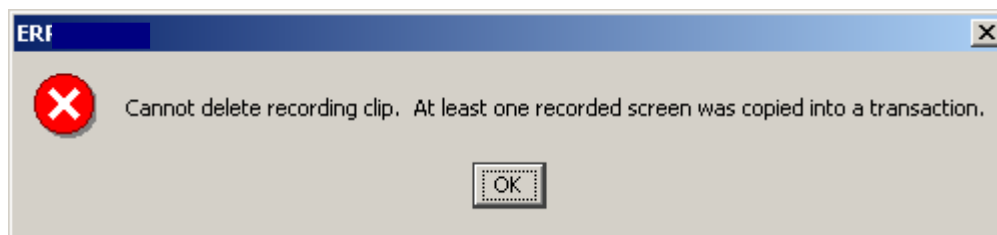


Figure 10: Delete Recording Clip

4. Otherwise, the selected recording clip will be deleted.

Create a Transaction

A transaction is a representation of a series of host screens that provides a mechanism for the navigation of input/output fields through the host application without operator interaction. When an existing transaction set is opened, a screen similar to the following is displayed:

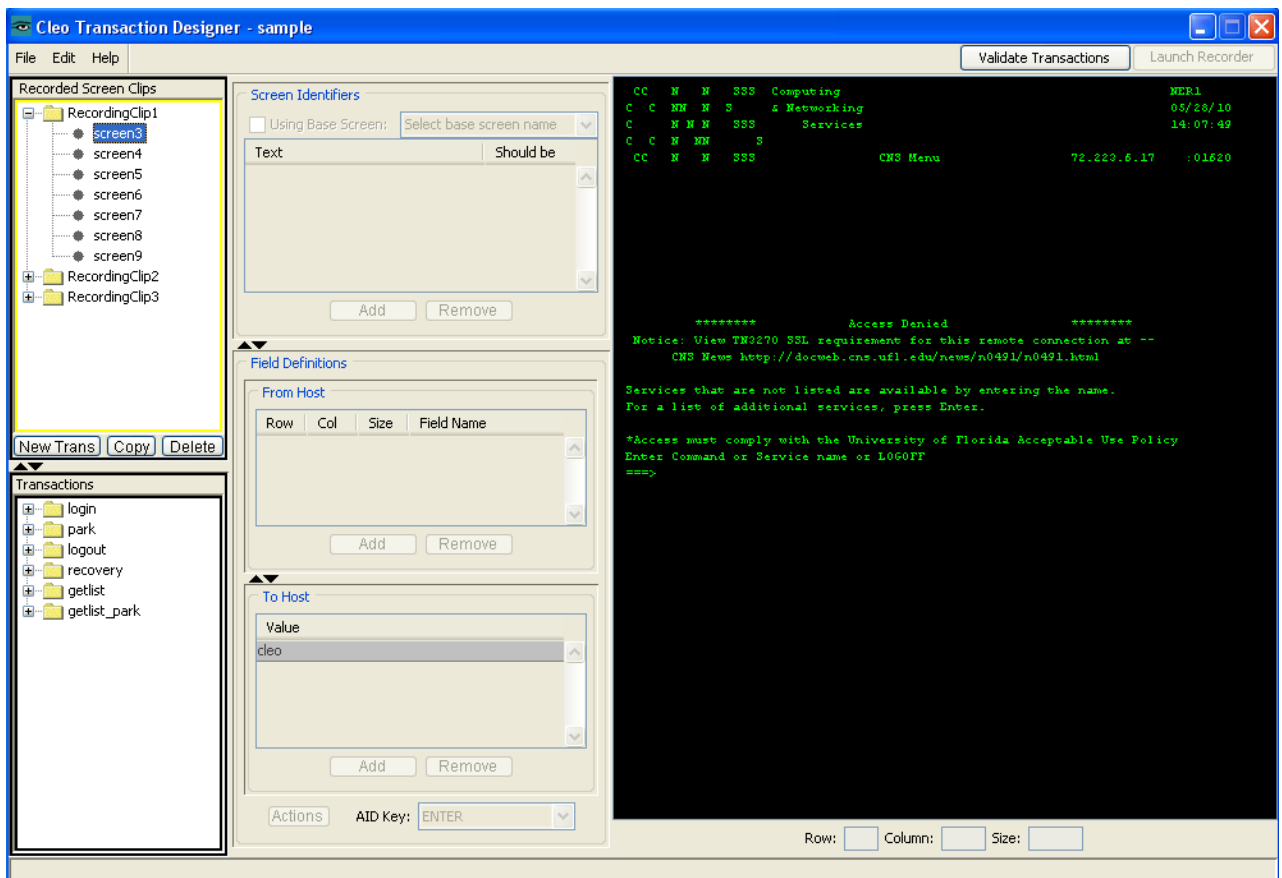


Figure 11: Transaction Creation Palette

The Transaction Creation Palette (TCP) is organized into six areas:

- **Recorded Screen Clips**-this area shows the names of all the recorded screens from either a previously recorded screen clip or a recording just completed.
- **Transactions**-this area shows the transactions that have been or are being created for this transaction set. It is the main work area when creating transactions
- **Screen Identifiers**-this area is where all screen identifiers are recorded. It shows the identifiers for the current screen being viewed
- **From Host**-this area is where all fields containing data coming from the host are specified.
- **To Host**-this area is where all fields containing data that is sent to the host are specified
- **Screen Viewing Area** is on the right and always displays the last screen highlighted (i.e. selected) under Recorded Screen Clips or Transactions.

In the Transaction section, there are four pre-defined transactions that may be defined within any transaction set: Login, Logout, Park and Recovery.

- Login and Logout:required for starting and ending a mainframe session.
- Park: Optional transaction that is ran after the Login transaction and can be used to place a session at a defined mainframe application screen so it is ready

for the next transaction. Frequently, this would be the first screen of the mainframe application.

- **Recovery:** the set of screens/actions to be executed when a transaction fails. The developer is responsible for specifying the screens within these transactions.

To create a transaction requires the following basic steps, which are explained in detail in the next few pages:

1. Create a new transaction using the New Trans function
2. Select (highlight) the recorded screen(s) to “Copy” into the transaction
3. Select the Screen Identifiers for each screen so it is uniquely identified
4. Define the From Host and To Host Fields in each screen

There are a number of features provided to simplify and expedite the creation of transactions, such as Blank Screen, Base Screen, Any Screen, Rename Screens, and Validate a Transaction. These features will be explained once the basic concepts are covered.

Create New Transaction

To create a new transaction: Click on **New Trans**.

The name given to the first user-defined transaction is “transaction1” and its companion, “transaction1_park”. The screens required to run the transaction are copied into “transaction1”. The screens required to **Park** the host application may be copied into “transaction1_park”.

The transaction that will be used when the processing of a customer call is ended requires consideration. The transaction can either:

- include logic in the transaction to get the session back to the **parked** state, or
- be accompanied by a companion transaction, **<transaction_name>_park**.

For example: Assume **transaction1** has been run at the request of the IR application.

The **transaction1_park** transaction contains screens needed to navigate back to the **parked** state screen, and is automatically run by the Transaction Processor when the IR application ends a customer call by releasing the session.

There are two advantages to using a companion **<transaction_name>_park** transaction:

1. The Cleo Transaction Processor can be processing the **<transaction_name>_park** transaction to get the session back to a **parked** state, quickly, while the IR application is gracefully exiting.
2. An IR application doesn't need extra logic to re-park the session, in case the caller chooses to make an early exit from a call.

NOTE: If a transaction contains a branch or rule resulting in one or more subsequent transactions to be run, there must be a companion **_park** transaction for each of the possible transactions that could be run. The main **<transaction_name>__park** would never be run in this situation.

In order to make the names of screens and transactions meaningful, they can be named using the Rename function in the Edit pull down menu.

In some applications, the screen returned by the mainframe application may be unknown (e.g. Any) or Blank. The next section shows how to add these screen types to a transaction.

Add Blank or Any Screen

To add a “Blank” or “Any” screen to a transaction:

1. Right-click on the appropriate transaction folder. Select **Add blank screen** (or **Add any screen**):

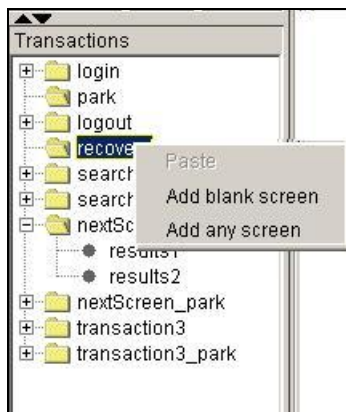


Figure 12: Add Blank Screen

2. The new screen is added to the end of the transaction.
3. To move the Blank or Any screen within the transaction, use the cut/paste process described on page 34.

Change Screen or Transaction Name

To change the name of a transaction, recorded screen, or screen within a transaction:

1. Select the transaction, recorded screen, or screen within a transaction to be renamed.
2. Select **Rename** under the **Edit** menu item.
3. A **Rename** dialog box is displayed for typing the new name. Enter the new name. Click **OK**.

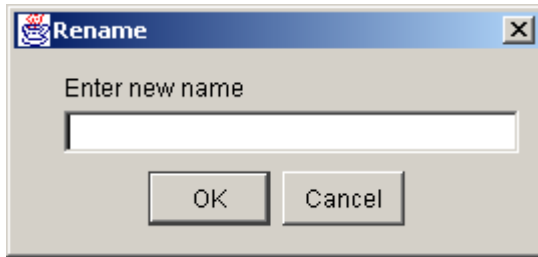


Figure 13: Rename Transaction

- The pre-defined transactions (*Login, Logout, Park, Recovery*) cannot be renamed.
- Names must be unique across recording clips as well as transactions.

Create Screen Definitions

Once screens are added to a transaction, they need to be identified with Screen Identifiers and have any From Host or To Host fields specified.

The panel below shows a list of transactions and the screens used in those transactions.

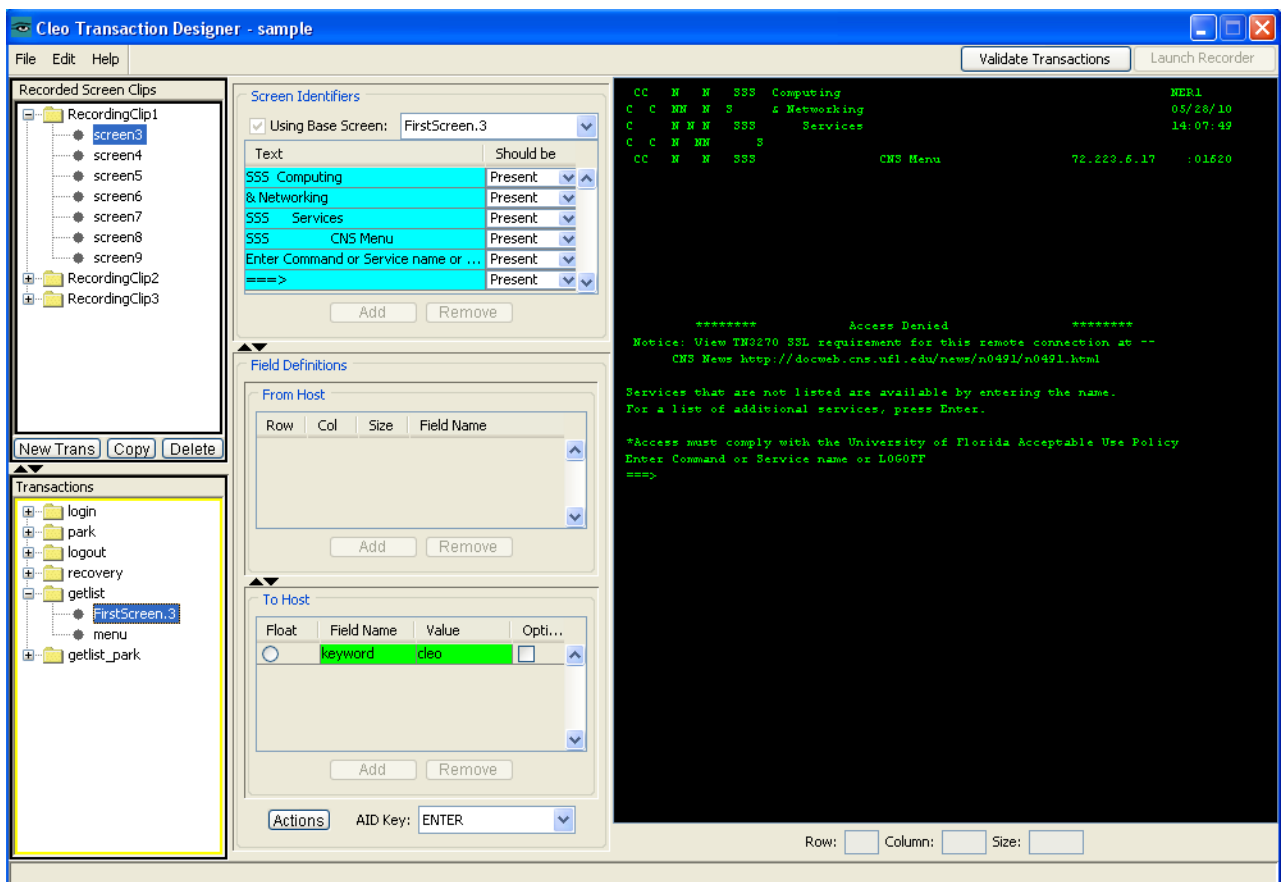


Figure 14: Screen Identifiers & Field Definitions

When a screen name is selected, the screen contents are displayed in the panel on the right as green text on a black background. Additionally, any input typed on the screen is displayed in the “To Host” table under “Field Definitions”. Also, the AID key used to navigate to the next screen is displayed.

Each screen must have at least one identifier. Screens can be identified using the following criteria:

- One or more areas on the screen containing text
 1. Swipe an area on the screen displayed at the right.
 2. Select the **Add** button in the **Screen Identifiers** box, shown in Figure 14. The contents of the highlighted area will be displayed under **Text**. Note: The start row and column, and the length or number of characters of the highlighted area will be displayed at the bottom of the screen.
 3. Associated with each identifier under the **Should be** column is a drop-down box containing **Present** and **Absent**.
 4. Select **Present** if the screen is to be identified by the text displayed under **Text**.
 5. Select **Absent** if the text as displayed should NOT be present on the screen at the designated location.
 6. Repeat Steps 1-5 until the screen is uniquely identified

Since screens may arrive from the mainframe in sections, it is preferable to have screen identifiers on several sections of the screen to assure the complete screen is present.

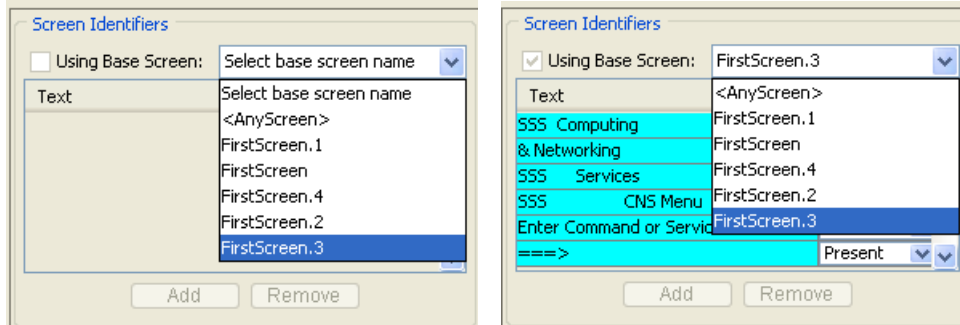
- **Blank:** screen contains no text
- **Any:** any screen content, including blank
- **Base:** A previously defined screen with the Base designation

To expedite the creation of transaction sets that use the same screen in several transactions, a Base Screen can be used to specify the screen identifiers for any transaction screen that contains exactly the same text as the **Base** Screen.

The use of a Base screen has several advantages. First, it eliminates the repetitive definition of screen identifiers. Second, it allows the screen identifiers to be changed for all screens based upon the Base screen with a single edit to the Base screen. Third, all the transactions predicated upon the Base screen can be displayed as described in List Screens Using a Base Screen.

All screens that have text screen identifiers specified are automatically created as Base Screens.

To use a **Base Screen**, first select the screen in the transaction, then use the pull-down box to select the base screen name to use for its identifiers. In the following example, “FirstScreen.3” is selected. The identifiers from “FirstScreen.3” are then displayed in the table of identifiers for screen1.2.



If the transaction screen is to be considered an **Any** screen, then the “<AnyScreen>” selection is made. When running a transaction, the Transaction Processor will accept the **Any** screen from the host regardless of what text is on the screen.

List Screens Using a Base Screen

To view a listing of all screens within the transactions which use the highlighted Base screen for its identifiers:

1. Highlight the **Base Screen**.
2. Select **Edit | List Screens Using Base Screen**.

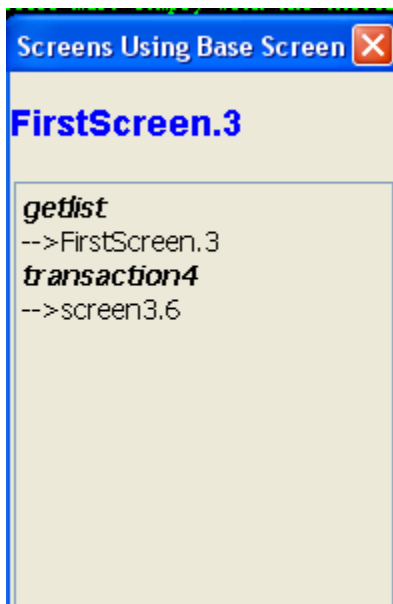


Figure 15: Screens using FirstScreen as Base

This feature is useful for understanding which transactions will be impacted by changes in the Base Screen they use.

Define Fields

Define Fields Received from the Host

Initially, the **From Host** box is empty, indicating no fields on the selected screen are to be “scraped”. Fields listed in the **From Host** box contain data that has been received from the host that can be requested by the IR application using the Transaction Processor *getOutput* function.

To add new **From Host** fields:

1. Use the mouse to highlight (swipe) an area on the host screen image.
2. The start row and column and length (i.e., number of characters) of the highlighted area will be displayed at the bottom of the screen.
3. Select the **Add** button in the **From Host** box, shown in Figure 14, to cause a row to be added to the **From Host** table, with the row, column, size and default field name (“field<n>”) specified.

To change the name of an existing **From Host** field:

1. Double click in the text box.
2. Type the new name in the displayed dialog box.
3. Click **OK**. If the name is not unique within the screen, an error box is displayed.

The values of all the **From Host** fields from each screen within a transaction are available to the IR application after a single successful run of the transaction.

Define Fields Sent to the Host

The input fields on a host screen are those areas into which data can be typed. When the screen recording is taken, the data that was typed into each input field is recorded and displayed in the **To Host** box. The field names are given the default value “field<n>”.

To change an existing **To Host** field name or value:

1. Double click on the field name or value in the appropriate box within the **To Host** panel.
2. Type the new name or value in the displayed dialog box.
3. Click **OK**.

To add a new **To Host** field (an input field that was not typed into at recording time) to the **To Host** list shown in Figure 14,

1. Use the mouse to highlight (swipe) the area on the host screen image.
2. Click **Add**. (If any part of the swiped area is not a valid input position on the host screen, then **Add** is not enabled.)
3. The field name and value are then displayed in the **To Host** box.

To designate that a **To Host** field be located at the initial cursor position on the host screen (which can vary):

1. Select the radio button in the **Float** column.
2. The text in this field will then be written at the initial cursor position on the host screen when first received from the host.

NOTE: Only input (unprotected) fields can be added to the To Host box. These fields will be populated with the field value and sent to the host unless the checkbox in the Optional column is checked. If a field is Optional, the Transaction Processor addInput function must be used to populate the field before sending to the host.

Highlight Input Fields

To display all input fields in gray, turn on Highlight Input Field Boundaries by selecting **Edit | Highlight Input Field Boundaries** from the menu. This will help determine where to swipe for **To Host** definitions. Select **Edit | Highlight Input Field Boundaries** again to hide input fields.

Figure 16 shows an example of a display screen with Highlight Input Field Boundaries turned on.

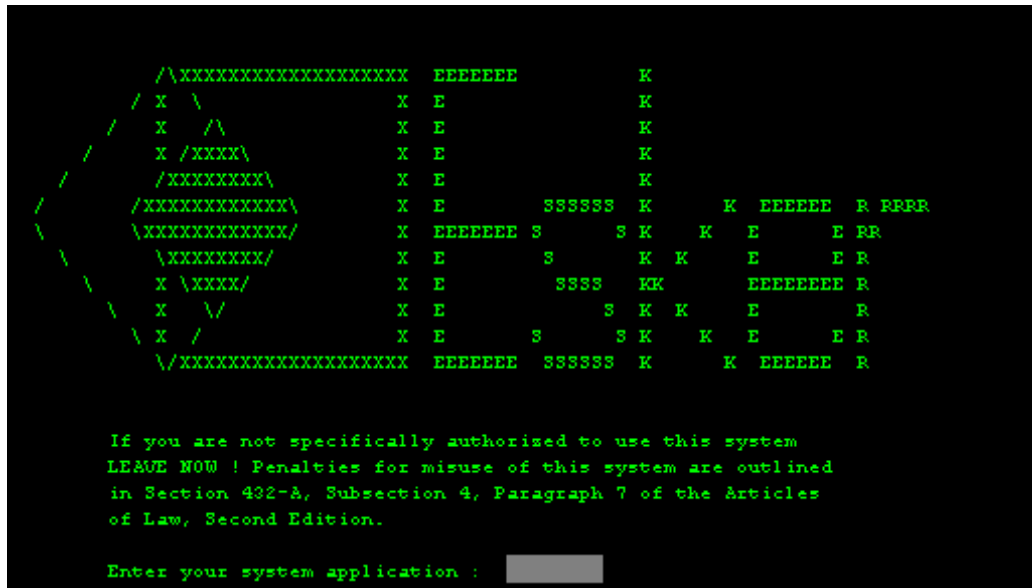


Figure 16: Sample screen with Highlight Input Field Boundaries

Define Table Driven Input Field Values

There are situations when data in a particular field to be sent to the host varies based upon the session. A typical example is logging in many sessions at system start up where unique logon ids and passwords are required for each session.

This data varies by session but is predetermined and fixed. This type of input is created using the Keyboard Macro capability in the Transaction Processor.

To denote that the field should be filled with variable data before being sent to the host, enter a number surrounded by square brackets, such as “[2]”, in the **To Host** field value column. The number indicates which column (in the keyboard macro table) contains the field's actual contents. (Refer to the *Cleo 3270 Plug-in Administrator's Guide* or the *Cleo TP Administrator's Guide* for information on how to define the keyboard macro table.)

The macro table will be accessed by the Transaction Processor at run time and the appropriate substitutions will be made. Note that there is a keyboard macro definition for each session that is configured.

For example, given the following keyboard macro table:

Session	Transaction Set	Host	Enabled	Keyboard Macros			
				1	2	3	4
1	sample	sample	<input checked="" type="checkbox"/>	loginid1	passw1		
2	sample	sample	<input type="checkbox"/>	loginid2	passw2		

Figure 18: Macro Table from Transaction Processor Configuration Web Utility

and the following **To Host** Definition:

Float	Field Name	Value
<input type="radio"/>	field1	[2]

Add Remove

Figure 19: "To Host" Definition (Transaction Designer)

the “[2]” will be replaced with “passw1” when using Session 1, and replaced with “passw2” when using Session 2.

Define Actions

Actions provide a means to add additional capability to a transaction.

- **Reset:** Powers the session on and off and runs the login transaction. This command can be added to any screen within a transaction and will be executed before the remaining screens in the transaction are processed. The reset command can be added to a user-defined recovery transaction to take the place of automatic recovery.
- **Capture:** Dumps screen contents to a file. This command is useful for added diagnostics.
- **Branch:** Runs a transaction automatically based on what screen arrives from the host. For more details on branching see Appendix C: Conditional Transactions on page 64.
- **Rule:** Runs a transaction automatically based on the value(s) of one or more **From Host** fields arriving on one or more screens from the host. For more details on using rules, see Appendix C: Conditional Transactions on page 64.

To define an **Action**:

1. Click on the **Actions** button located below the **To Host** panel.

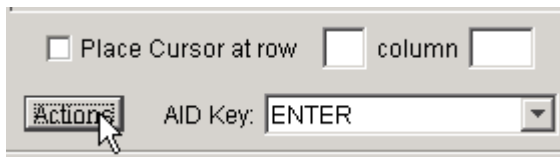


Figure 20: Define Action

To define a **Reset** command:

1. Click the **Reset** tab.
2. Click **Add**.

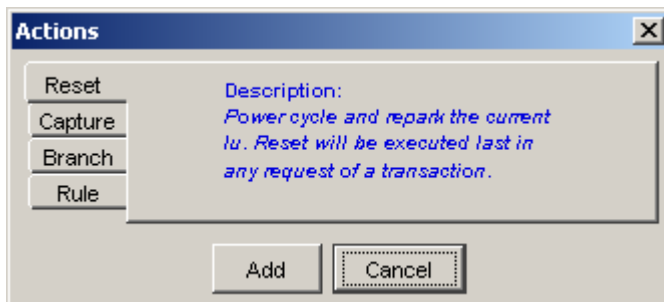


Figure 21: Define Action "Reset"

To define a **Capture** command:

1. Click the **Capture** tab.
2. Specify the full path name for the file to contain the captured screens.
3. Click **Add**.

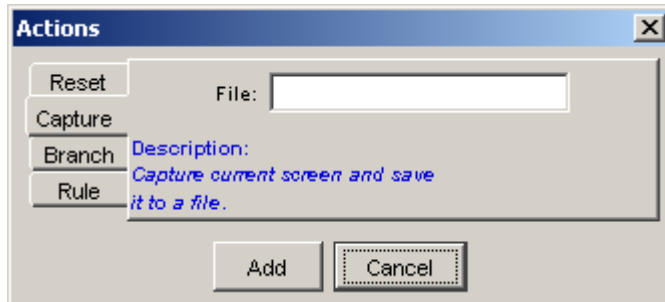


Figure 22: Define Action "Capture"

To define a **Branch** command:

1. Select the screen from which the branch will occur (Note: Branch can only be defined on the last screen in a transaction).
2. Click the **Branch** tab.
3. Click **Define Branch**.

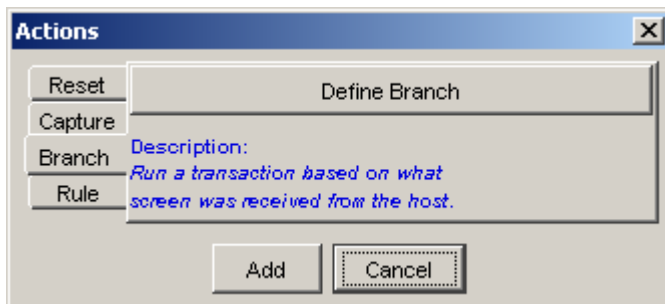


Figure 23: Define Action "Branch"

4. Enter possible return screens and associated transactions to run next by clicking on the down arrows and selecting the screen and transaction names.

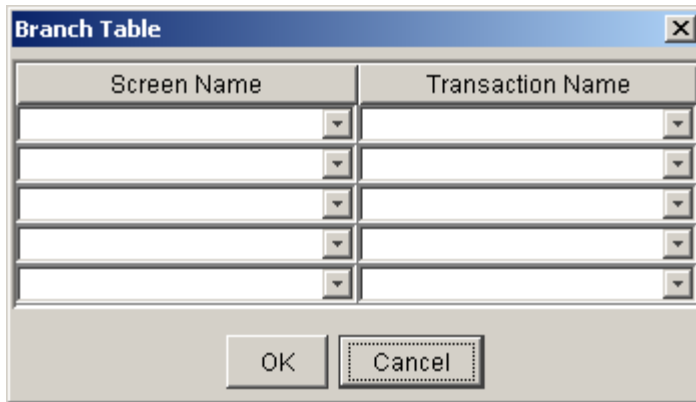


Figure 24; Branch Table

5. Click **OK**.
6. Click **Add** displayed in Figure 23.

To define a **Rule** command:

1. Select the screen from which the rule will occur (must be the last screen in a transaction).
2. Click the **Rule** tab.
3. Click **Define Rule**.

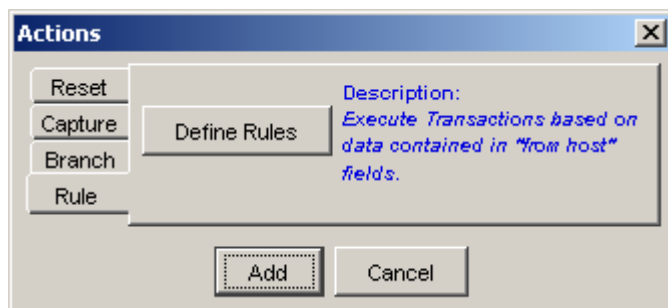


Figure 25: Define Action "Rule"

4. Build all possible rules

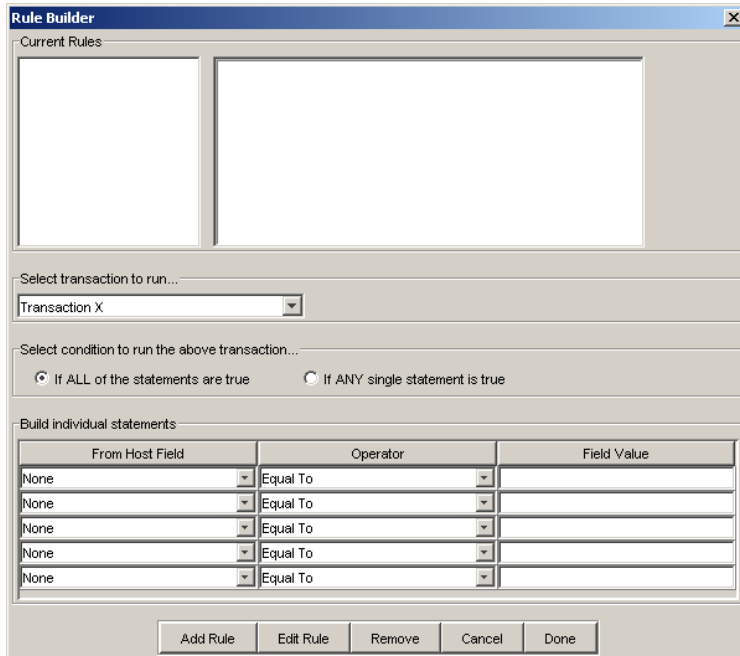


Figure 26: Rule Builder

5. Click **Add Rule**.
6. Click **Add** from Figure 25.

After adding a **Reset**, **Capture**, or **Branch** action, it will appear as a **To Host** field with field name “_COMMAND<n>” and value, as follows:

[@RESET], [@CAPTURE|<path>], or
 [@BRANCH|<screen1>|<transaction1>|...<screenN>|<transactionN>

After adding a **Rule** action, it will appear as a **To Host** field with field name “_COMMAND<n>”. Double clicking on the field value starting with **@RULE** will display the rule(s) defined, as shown in Figure 27: Example of Rule in sample.

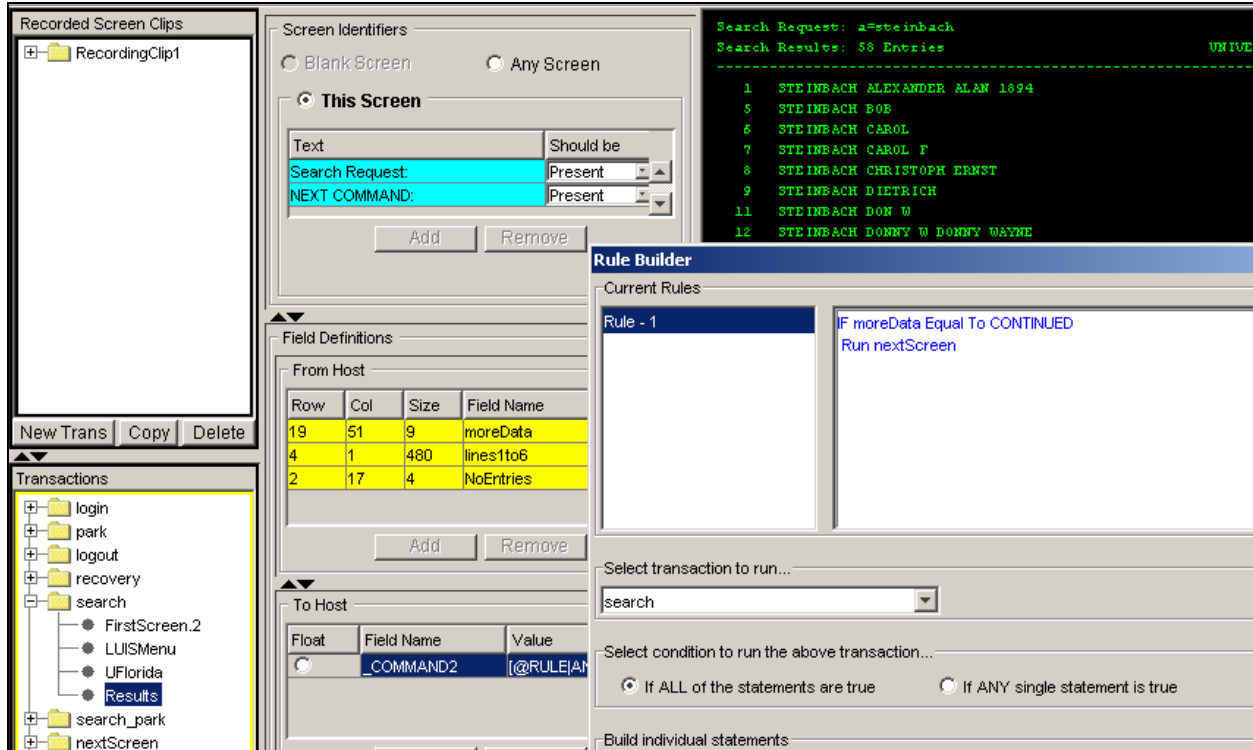


Figure 27: Example of Rule in sample

This example shows the sample transaction set containing the “search” transaction which has a rule defined in the screen named “Results”. The rule states that if the value of the **From Host** field named “moreData” equals CONTINUED automatically run the “nextScreen” transaction to page forward to the next screen of results.

For more information on creating Rules and Branches see Appendix C: Conditional Transactions.

Example of a complete screen definition:

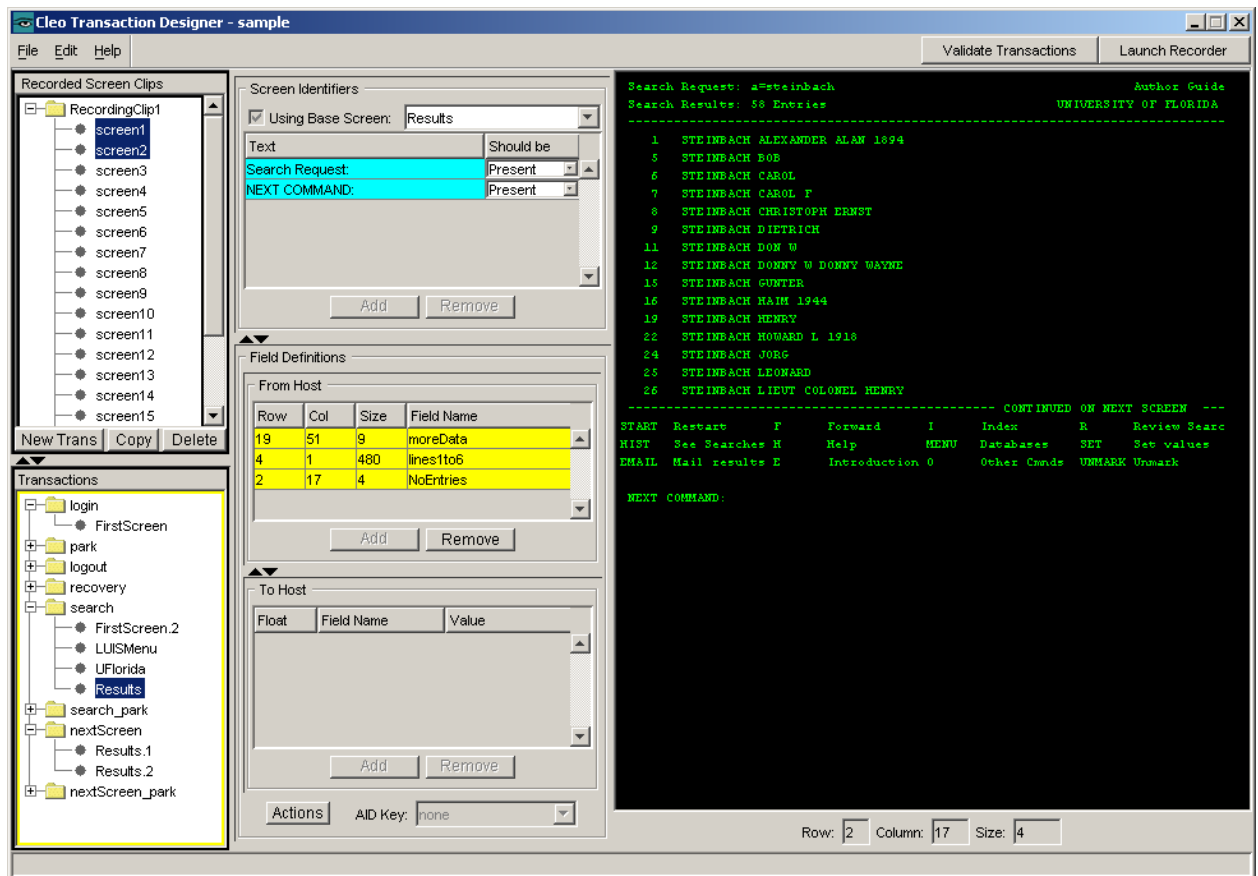


Figure 28: Sample Screen Definition Completed

Duplicate Screen Definitions

Often there is a need to use the same screen several times in transactions. To avoid repeating the screen identification process for the same screen multiple times, copy the screen and all its definitions and paste it anywhere in an existing transaction.

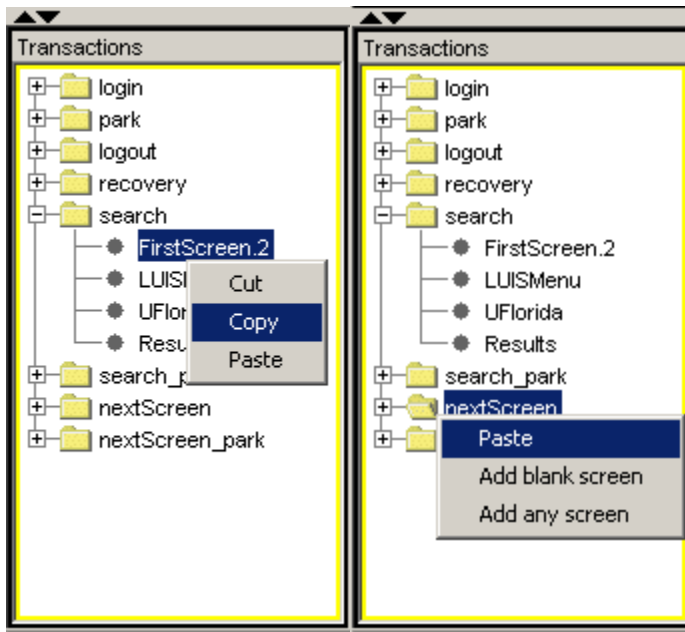


Figure 29: Copy and Paste Screens

To copy and paste screens:

1. Right-click the screen to copy and select **copy**. Select **cut** to delete the screen from its current location in the transaction.
2. Position the cursor where you want to insert the copied screen. (For example, left-click on a transaction name [or screen name] to highlight it. The insert will occur after the highlighted field.)
3. Right-click on the highlighted field and select **paste**.

The copied screen and all its fields and identifiers will be added after the highlighted transaction (or highlighted screen) and given a unique name.

If the screen that is being copied contains either a rule or a branch then the following dialog will appear:

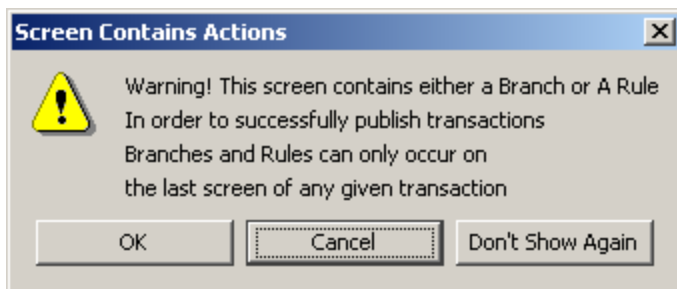


Figure 30: Screen Contains Actions

This is just a warning to let you know that Rules or Branches can only occur on the last screen of a transaction. If you click **OK** and paste a screen containing a Rule or Branch into a position other than the last in the transaction, when running **Publish**, you will be prompted to fix the problem. This causes the Rule or Branch to be removed so that a successful publish can take place. Click **Don't Show Again** to avoid repeatedly seeing this dialog.

Publish the Transaction Set

When the host transactions have been defined, the **File | Publish** menu item is used to create the files to be used by the Transaction Processor or Cleo 3270 Plug-in on the IR system. A dialog box similar to the following will display after a successful publish:



Figure 31: Host Transactions Successfully Published

If any screen definitions, such as missing identifiers, are incomplete, an error dialog box is displayed.

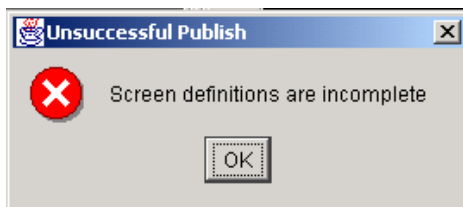


Figure 32: Screen Definitions Incomplete

In the case of a partial Publish, a dialog box will be displayed identifying any transactions that could not be published because one or more screens are missing identifiers. Refer to Incomplete Publish on page 48 for more information on this and other problems that can occur.

Transfer the Published Transaction Set

All files with the “xml” extension in the Transaction Designer’s *trans* directory (default: C:\Program Files\CleoTD(trans)) must be moved to the *trans* directory of the Transaction Processor or *apps* directory of the Cleo 3270 Plug-in. There are several methods to complete the move:

- In Windows File Explorer, copy files to the appropriate directory for use by the Transaction Processor.

- Transmit the files to the IR system using an FTP. If necessary, telnet to the IR system to move files into the appropriate directory for use by either the Cleo 3270 Plug-in or the Transaction Processor.
- Use the built-in transfer feature to move files to the appropriate locations. The use of this feature requires that a successful publish must have taken place.

To access the transfer feature:

1. Select **File | Transfer XML** from the menu.
2. From the following dialog, provide a name so that the settings can be referenced again at a later time.

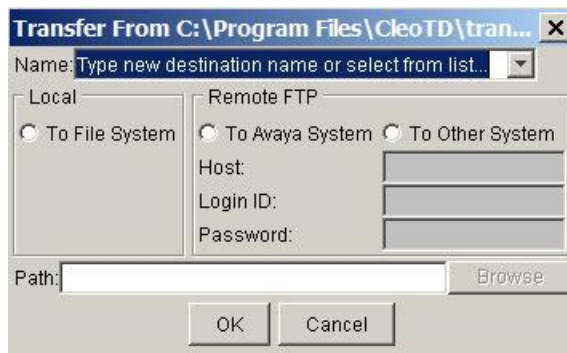


Figure 33: Transfer XML

3. Depending on where (to which system) the move will be made:
 - Select **To File System** and browse to the **Path** that exists on the same machine that the Transaction Designer is running.
 - Select **To Avaya System** and enter the appropriate Host, Login ID, and Password to automatically transfer the files via FTP to the /cleo3270plugin/apps/<transaction set name> directory on the requested Avaya (CONVERSANT) system.
 - Select **To Other System** and enter the appropriate Host, Login ID, and Password, and Path to automatically transfer the files via FTP to the provided directory on the requested remote system.
4. Click OK.

NOTE: Before attempting the transfer, write permissions must be set on all directories that will be accessed. The transaction set folder is created if it doesn't already exist.

Save Transaction Set under Different Name

To save the Transaction Set under a different name, simply select **File|Save As...** A dialog box will appear.

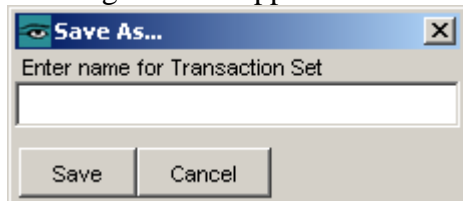


Figure 34: Save As

Enter the new name for the Transaction Set and click the **Save** button. The entire application will be saved under the new name. If you attempt to save a Transaction Set under a name that already exists, you will see the following message...

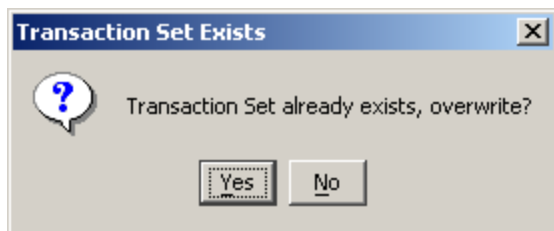


Figure 35: Transaction Set Exists

If you want to overwrite the information contained in this transaction set, click **Yes**.

Run the Validator

To test “published” transactions, the Transaction Designer offers a Validator feature. This feature is accessible from the Validate Transactions button in the upper right-hand corner of the TD screen. This feature requires a host connection for validation.

To test published transactions, perform the following:

1. From the Transaction Designer screen, select the transaction set you wish to test using **File|Open**. In the following figure, we are selecting the transaction set **sample**:

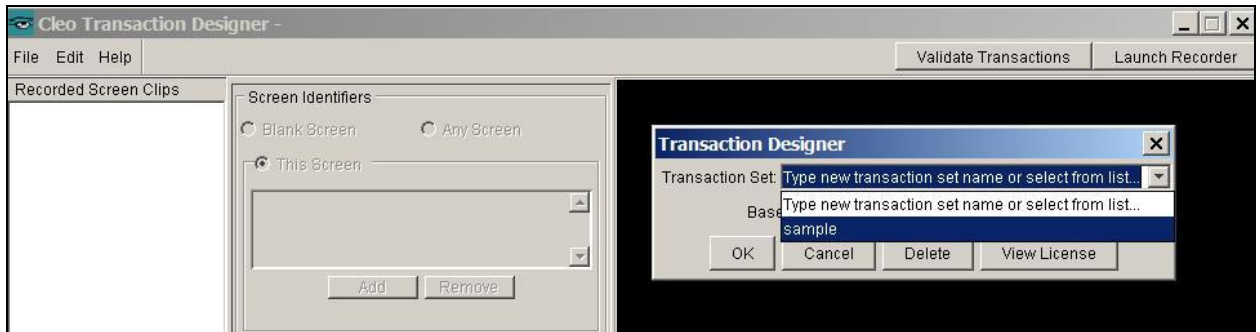


Figure 36: Select Sample to Validate

2. Next, click **Validate Transactions**.
3. The Validator screen then displays, listing the published transactions for the transaction set (e.g., **sample**):

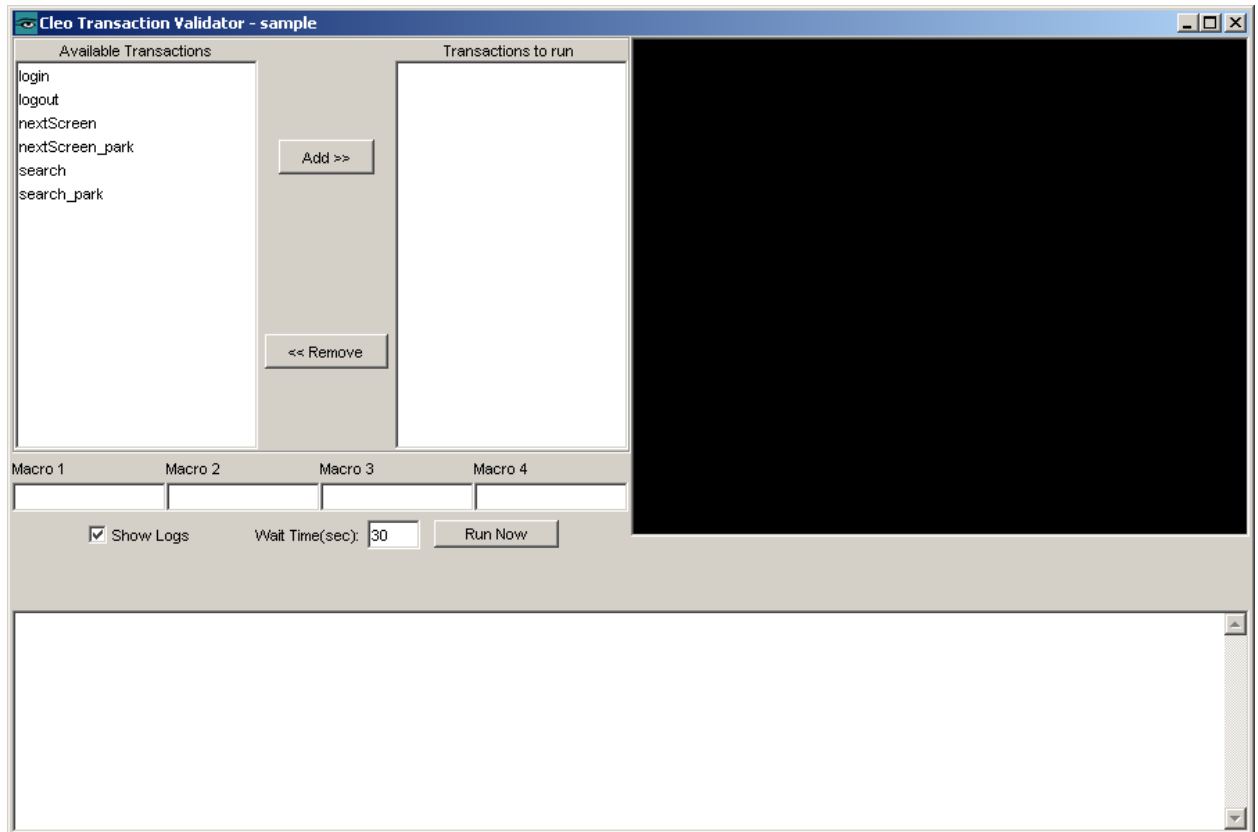


Figure 37: Initial Validator Screen

4. Highlight the first transaction, login. (Start with the login transaction first to navigate to the parked screen.) Click **Add**. The login transaction will move from the *Available Transactions* field to the *Transactions to run* field. Click **Run Now**.

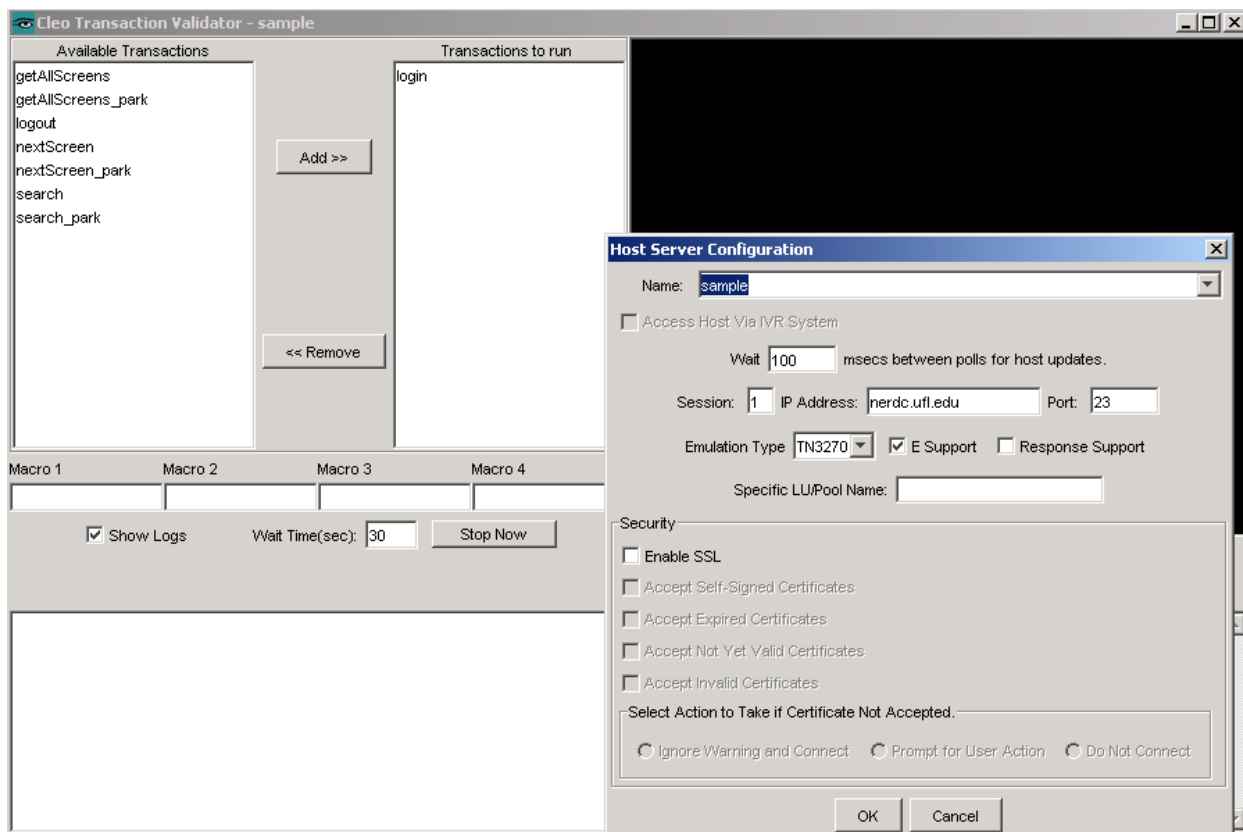


Figure 38: Host Server Configuration Screen

Since the Validator accesses a live host connection, you will need to provide host information before continuing. Provide the host name and IP address, and check *TN3270E Support* if applicable. Other fields contain default values; if necessary, change these to fit your particular situation. See section *Access the Screen Recorder* for more information.

If using a terminal screen size other than 24 rows by 80 columns (mod2), enter the session number <n> corresponding to the sess<n>.zcc file that was changed when using the TD recorder. See “Configuring for Alternate Screen Sizes” in the section, *Access the Screen Recorder*.

Click **OK**.

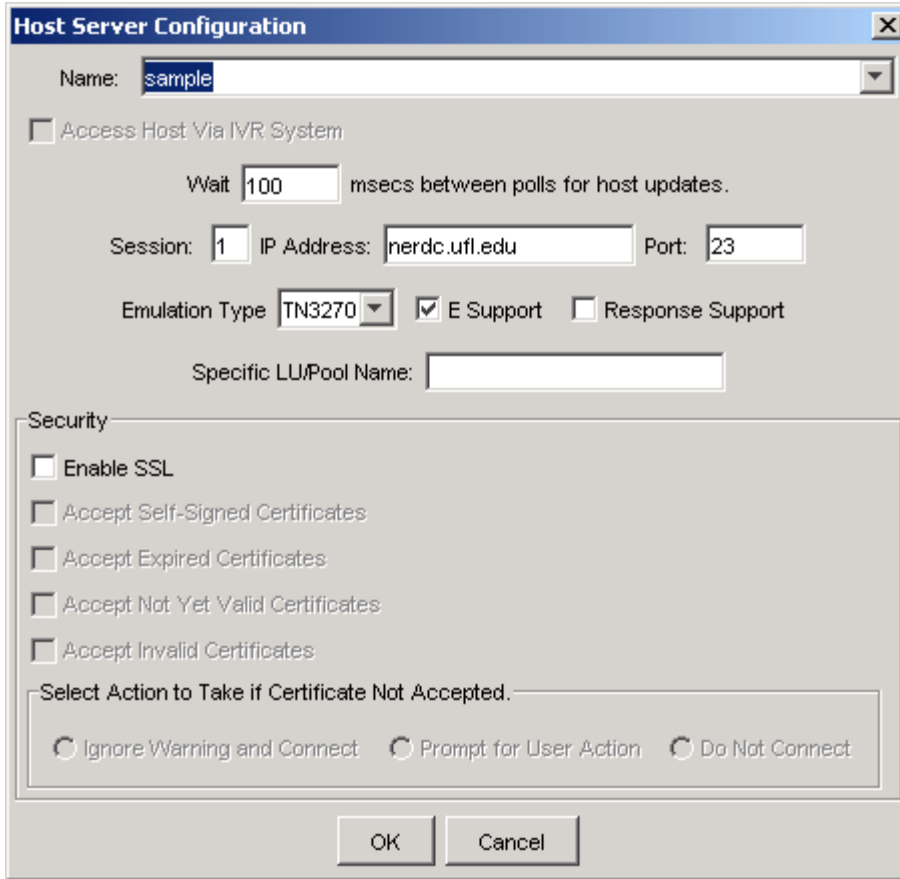


Figure 39: Host Configuration for sample

NOTE: *You'll only need to provide the host information once (i.e., on login) during this validation session.*

5. The Validator tests the first transaction (*login*), and then provides the test results on the bottom half of the screen. A return code of zero (*rc=0*) denotes success. As the transaction is run, screens are displayed to the right:

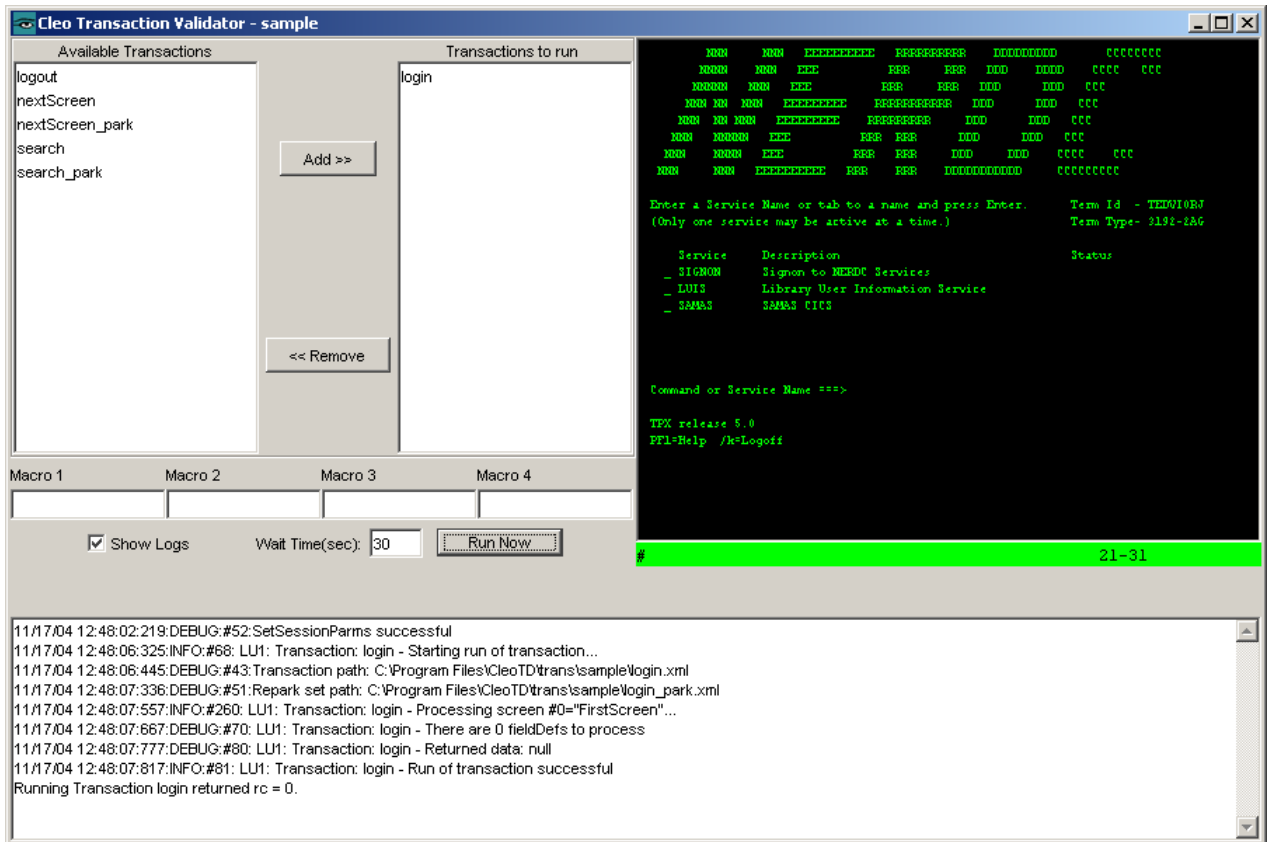


Figure 40: Successful Validation

NOTE: *If you receive an initial return code stating “Wrong screen detected” (rc=-103), it mostly likely reflects a delay in the connection to the host. Wait momentarily and then try running the transaction again.*

6. To continue, highlight *login* and move it back to the *Available Transactions* column by clicking **Remove**.
7. To test more transactions:
 - a. select (i.e., highlight) one or more transactions in the *Available Transactions* column,
 - b. click **Add** to move them into the *Transactions to run* column, and then
 - c. click **Run Now**.

When the test on the transaction(s) is complete, move them back to the *Available Transactions* column by highlighting them and clicking **Remove**.

NOTES:

- Test transactions in the proper order of their expected execution.
- Remember to test the corresponding “park” transaction following operational transactions (e.g., test “search_park” after testing “search”). This will assure a return to the proper screen location.

Sample screen sequence:

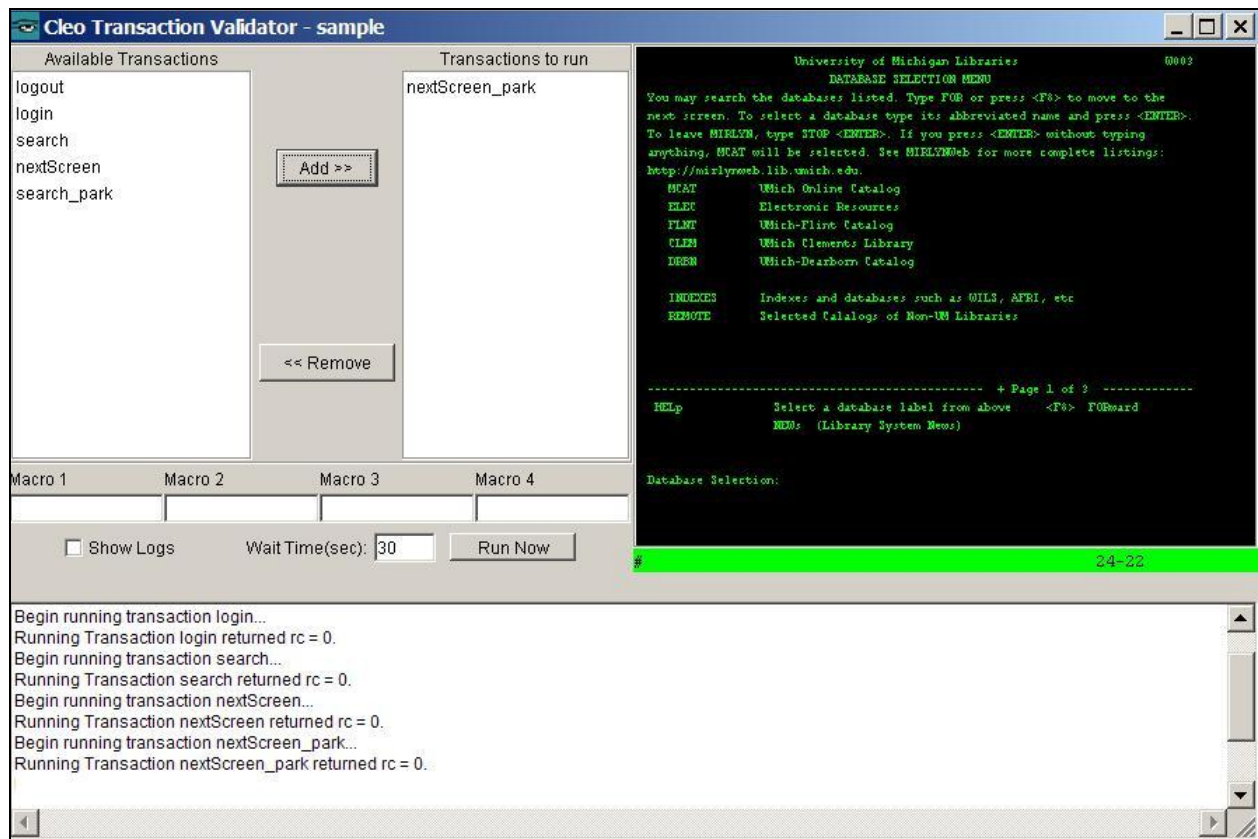


Figure 41: Validation Messages: Run Sequence

This screen illustrates the following sequence:

1. Validate login:
 - a. **Add** login
 - b. **Run Now**
 - c. **Remove** login
2. Validate search:
 - a. **Add** search
 - b. **Run Now**
 - c. **Remove** search

3. Validate nextScreen:
 - a. **Add** nextScreen
 - b. **Run Now**
 - c. **Remove** nextScreen
4. Validate nextScreen_park:
 - a. **Add** nextScreen_park
 - b. **Run Now**
 - c. **Remove** nextScreen_park

Note that the last validated transaction (a park transaction) returns you to the initial screen.

Show Logs

Uncheck this box if you do not want to display the more detailed log messages. These log messages can help in tracking down problems when running a transaction.

Wait Time

Specify the amount of time in seconds to wait for a screen from the host after an AID key is sent. This corresponds to the *Timeout* configuration parameter used by the Transaction Processor. This value can be changed before running a transaction to help determine the best value to use for the Transaction Processor.

Specify Macros

If your application will be using macros to define variable data, you must explicitly type the value of these macros in the Validator. This is necessary because the Validator does not have access to the macro values entered using the web-based TP Configuration tool. Refer to *Define* on page 27, for details.

Specify the macro values on the Validator screen to mirror the values recorded in the web-based TP Configuration tool. For example, if macro1 = *loginid1* and macro2 = *passw1*, type these values in the corresponding Macro fields on the Validator screen as shown in Figure 42: Using Macros in Validator.

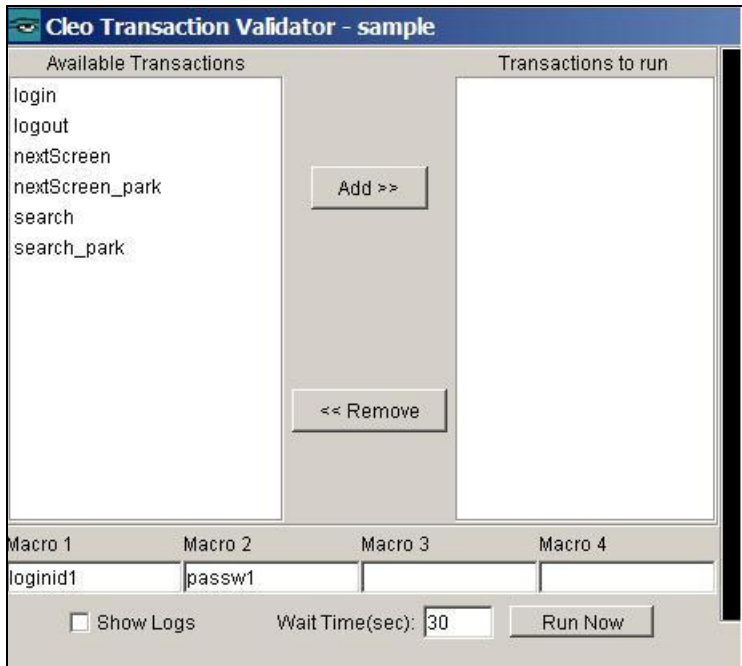


Figure 42: Using Macros in Validator

Menu Bar Descriptions

File Menu

Open: Open a previously existing transaction set. A dialog box is displayed where the transaction set name is entered.

Save: Save all work completed so far to the open transaction set's files. This includes recorded screen images, screen definitions, and transactions.

Create Map: Create and save the file TransMap.htm to the local disk in the transaction set's directory. This file contains a list of all host transactions and screen definitions currently defined within the transaction set. The following is displayed after successfully creating a map while the transaction set "sample" was open:



Figure 43: Create Map

See Appendix B: Sample Transaction Map on page 57 for an example of a transaction set map created for the sample application.

Publish: Create and save to the local disk drive all transaction files that can be run by the Cleo 3720 Plug-in or the Transaction Processor. This includes any serialized objects as well as XML files. All screens that are part of a transaction must have screen identifiers associated with them, or an error will be displayed. All files with the "xml" extension in the transaction set's directory (default: C:\Program Files\CleoTD(trans) must be moved to the Cleo 3270 Plug-in *apps* directory or Transaction Processor *trans* directory. See *Publish the Transaction Set*, page 35, for more details.

Transfer XML: Move the published XML files to the appropriate directory of the Transaction Processor or Cleo 3270 Plug-in. See *Transfer the Transaction Set* on page 35.

Exit: Terminate utility, closing any opened files. The user is prompted whether to save any changes made during this session.

Help Menu

About: Displays version, copyright information, and license information (Host ID, serial number, and expiration date).

Edit Menu

Rename: Renames a user-defined host transaction or a screen within any transaction that has been highlighted. A dialog box is displayed for typing the new name.

Delete Recording Clip: Removes the selected recording clip from the Recorded Screen Clips list.

List Related Screens: Displays a dialog that lists the screens within the host transactions that originated from the selected recorded screen.

Highlight Input Field Boundaries: Highlights input fields for the displayed screen in gray.

Appendix A: Troubleshooting

This appendix lists:

- General Errors
- Recorder Errors
- Restrictions

General Errors

Null Null

This message occurs at the time of initial installation if the system is not rebooted before attempting to view license information:



Figure 44: Null Null message

Reboot the system and try again.

Incomplete Publish

When selecting Publish under the File menu, this error indicates that one or more screens listed in the **Host Transactions** panel do not have screen identifiers defined:

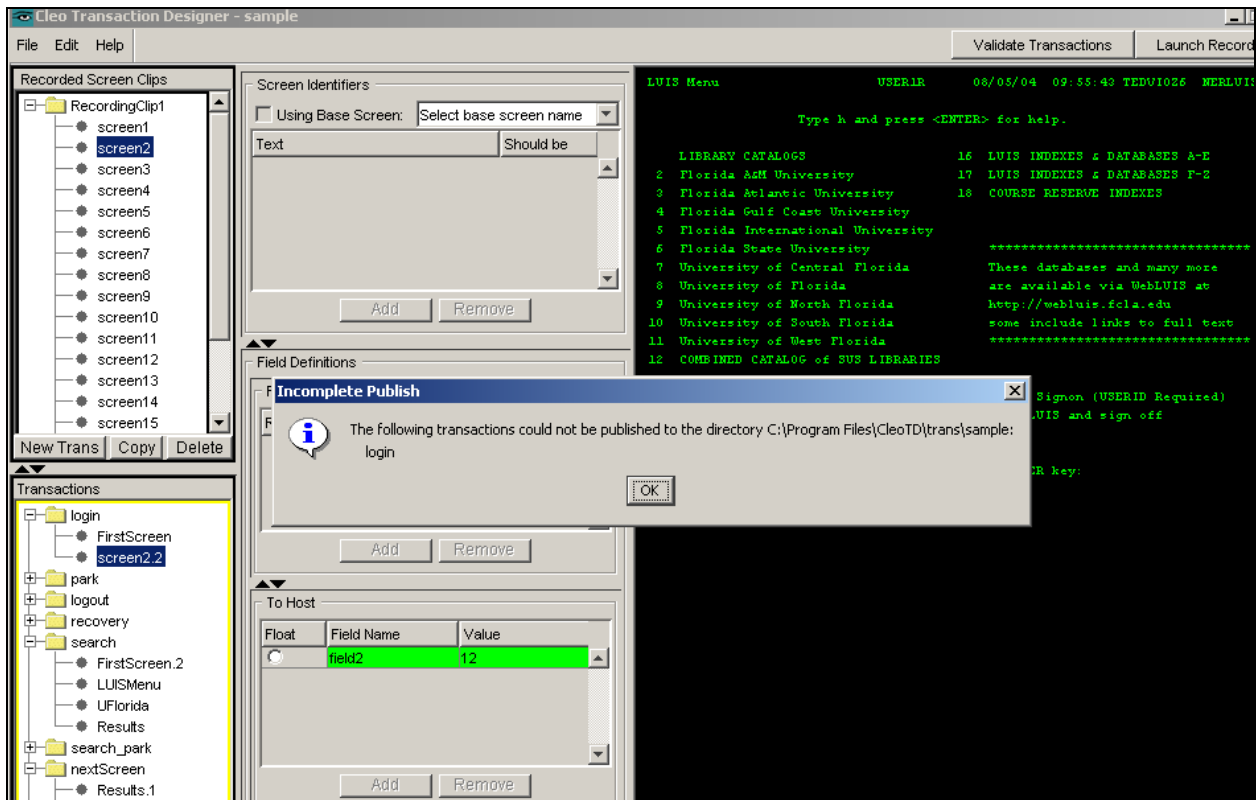


Figure 45: Incomplete Publish

There are no rows displayed in the box under “Screen Identifiers” for screen2.2, highlighted in the **Transactions** panel. A base screen can be selected, or a screen identifier added. All screens in the **Transactions** panel must have at least one screen identifier defined before the Publish function can complete successfully.

Screen Contains Actions

When selecting Publish under the File menu, this error indicates that the screen contains either a Rule or a Branch, which is not allowed. A Rule or Branch can only exist on the last screen of a transaction. Clicking on the **Fix Problem** button will automatically remove the action for you.

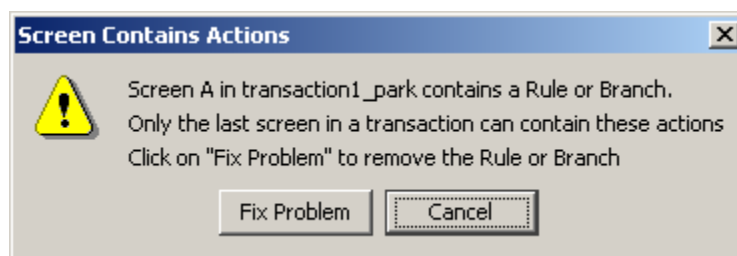
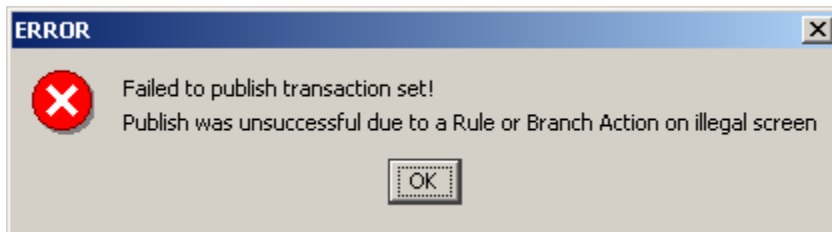


Figure 46: Screen Contains Actions

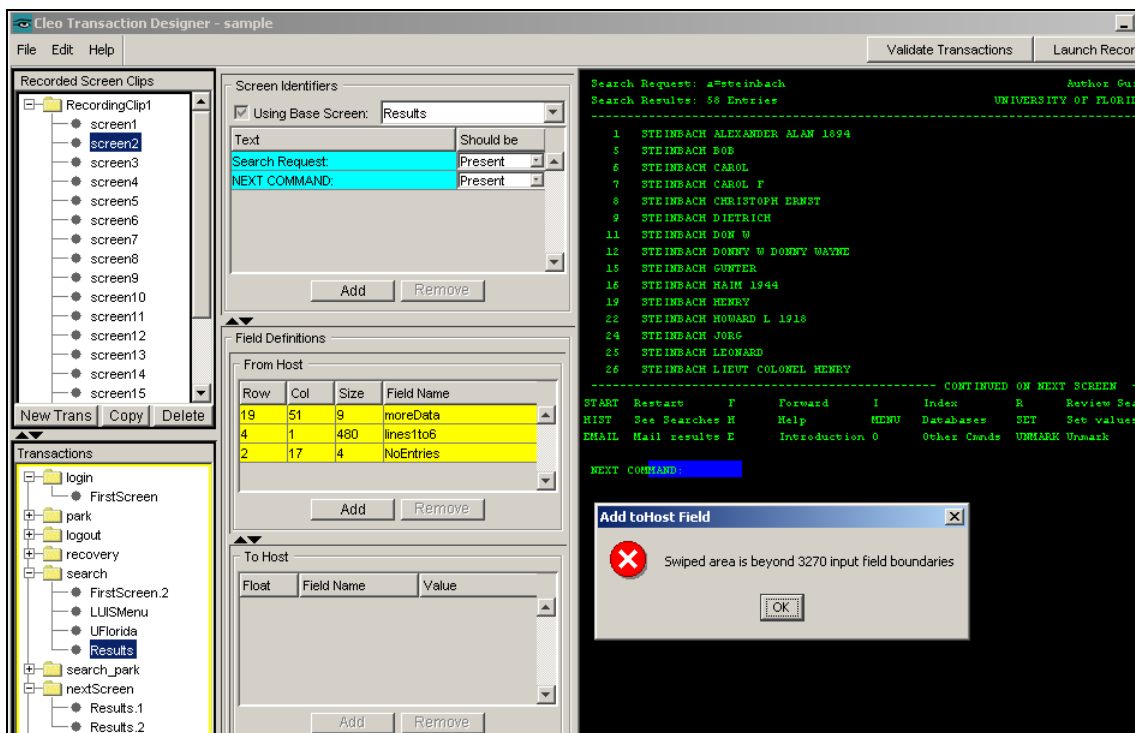
If the above actions are not removed or fixed, the following dialog will appear at publish time:

**Figure 47: Failed to publish transaction set!**

To fix this issue, simply republish and click the Fix option when prompted. All of the rules or branches in violation will then be removed.

Swiped area is beyond host input field boundaries.

The following figure shows that an area has been swiped on the line containing “USERID ==>”. An error occurs when clicking the **Add** button under the **To Host** panel:

**Figure 48: Swiped Area Beyond Field**

This error indicates that the area swiped spans more than one field on the host application screen. Try swiping a smaller area and clicking **Add**. The **Highlight Input Field Boundaries** function under **Edit** can be used to display the input fields for the displayed screen.

Sum of field lengths returned from host cannot be > 4000.

This error occurs when an attempt is made to define **From Host** fields **on one screen** whose lengths total more than 4000. The figure below shows that a large area of the host screen has been swiped.

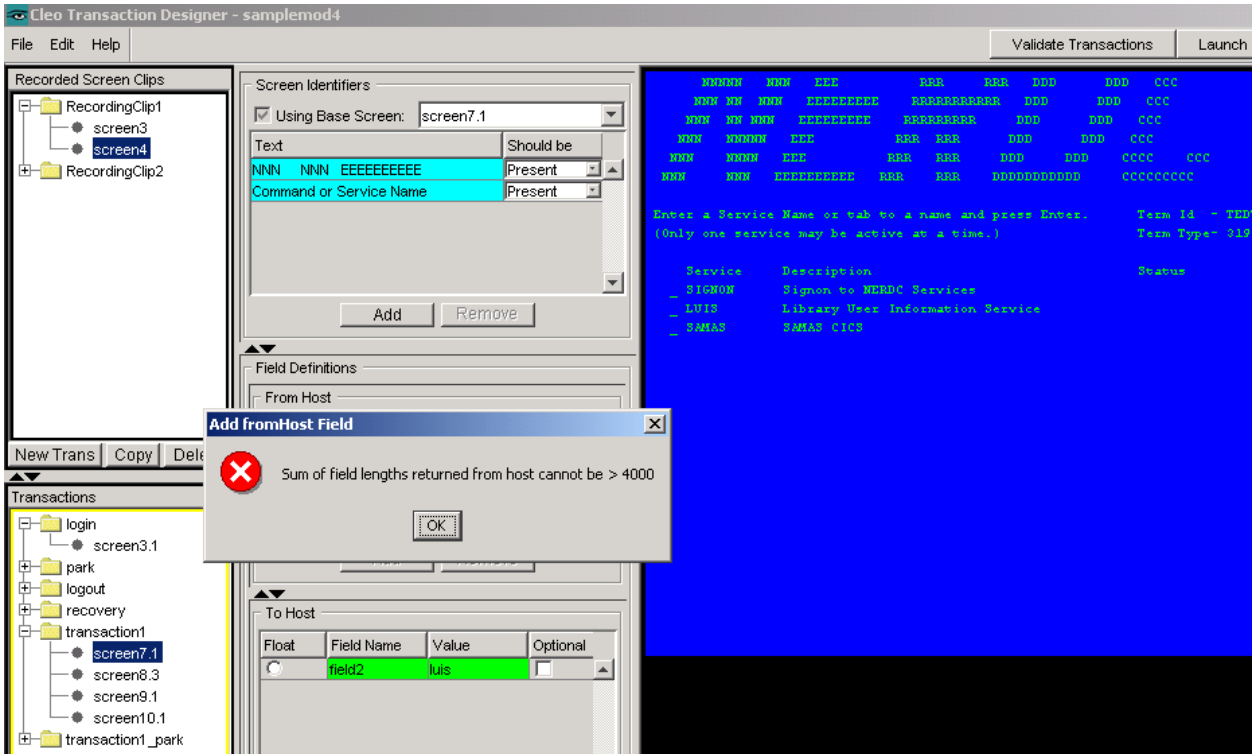


Figure 49: Large Screen Area Swiped

If other From Host fields have been defined in any other screen within the transaction, then it is likely that more than 4000 characters will be requested.

The sum of fromHost field lengths is > 4000 in host transaction "<name>".

If a Publish is attempted and the following warning occurs, the length of all From Host fields defined in the named transaction exceeds 4000 characters. The From Host field definitions specify areas that are to be scraped from the host screen and returned to the IR script.

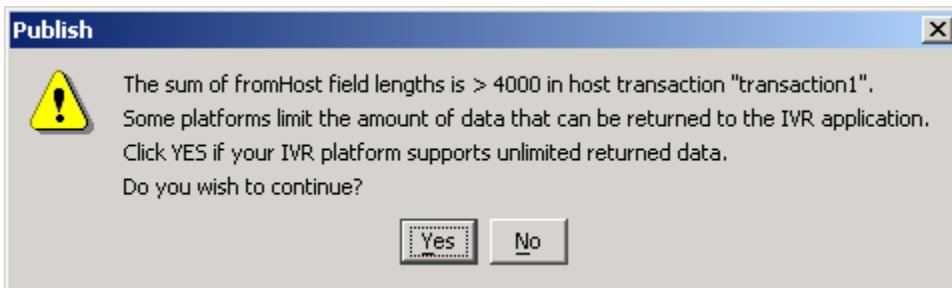


Figure 50: Sum of Host Fields > 4000

To correct this, check the rows in the **From Host** panels for each screen in the host transaction named in the warning message, and remove those fields that are not needed, or re-swipe a smaller area and click **Add** to re-specify the field definition.

No login transaction has been defined.

If no screens have been copied to the “login” transaction folder, the following error will occur when selecting Publish from the File menu. At least one screen must be copied from a recording clip to the “login” transaction. The Cleo Transaction Processor uses this screen to determine if the session is in a PARKED state, ready for use by an IR application.



Figure 51: Login Transaction error

Overlapping Fields

The following message will appear if a screen identifier, from host field, or to host field is defined whose location falls within or overlaps with a previously defined screen identifier, from host field, or to host field respectively.

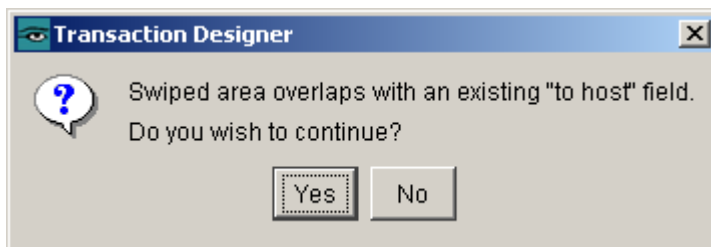


Figure 52: Swiped area overlaps with an existing “to host” field

Click **Yes** only if an overlapping field is desirable. Otherwise, click **No**.

Duplicate Fields

The following error will be displayed if a screen identifier, from host field, or to host field is defined whose location and length is defined to be identical to a previously defined screen identifier, from host field, or to host field respectively.

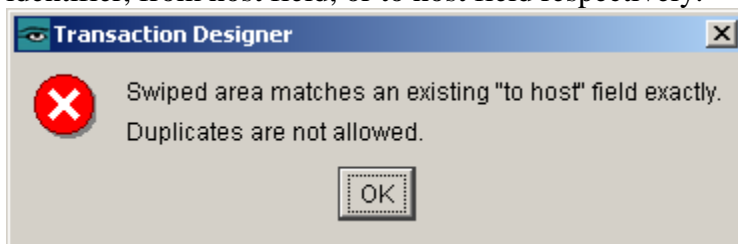


Figure 53: Duplicate Fields Error

Click **OK** to close the dialog.

Rename Failed

One of the following error messages will be displayed if user attempts to rename a from host field, to host field, screen, transaction, or recorded screen using Special XML Characters. These Characters include: Single Quote, Double Quote, Less Than, Greater Than, and Ampersand.

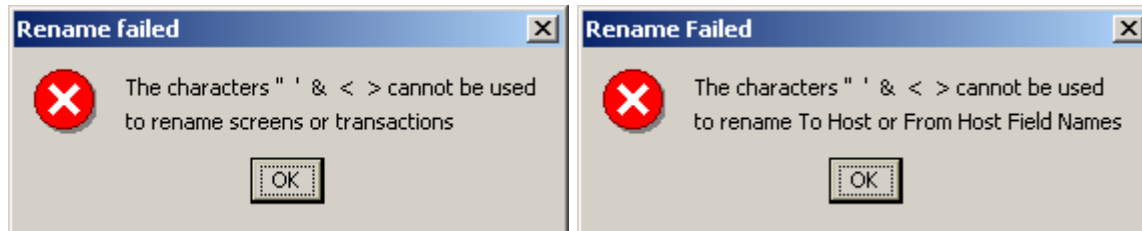


Figure 54: Rename failed

Field Definition Rename Failed

If the user attempts to rename a **To Host** or **From Host** field to a name that already exists within that transaction, the following error will appear.

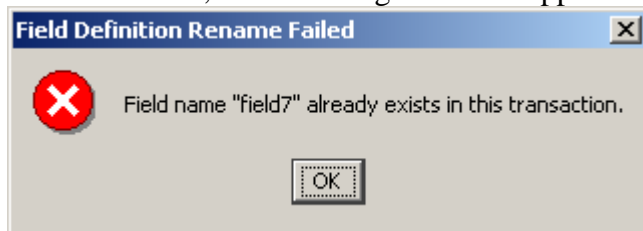


Figure 55:Field Definition Rename Failed

A screen that is used in a rule cannot be renamed

If the user attempts to rename a screen that is used in either a rule or branching transaction, the following dialog box will appear:

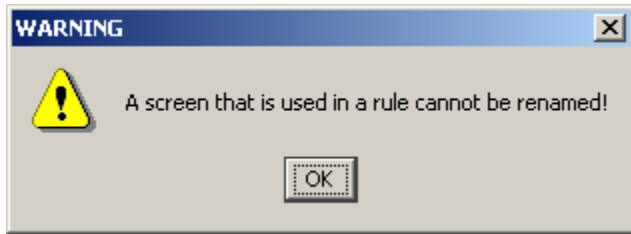


Figure 56: Cannot rename a screen used in a rule

In order to rename this screen the rule or branch associated with this screen must be deleted.

A screen that is used as a base screen cannot be deleted

If the user attempts to delete a screen that is used as a **Base Screen** by other transaction screens, the following dialog box will appear:

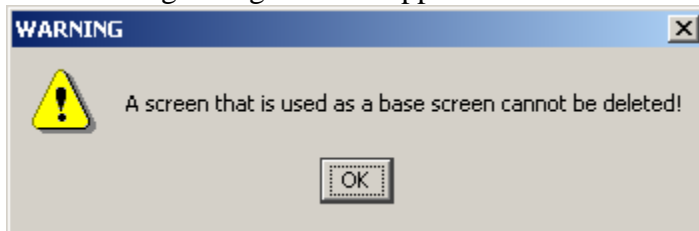


Figure 57: Cannot delete Base Screen

Corrupt files cannot be read

The extras.htd file contains important information pertaining to the relationships between the recorded clips and the transactions. If this file is corrupted or deleted, these relationships will be lost, although the transactions and recorded clips will still be fully functional. Upon opening a transaction set with a corrupted or missing extras.htd file, the following message will appear.

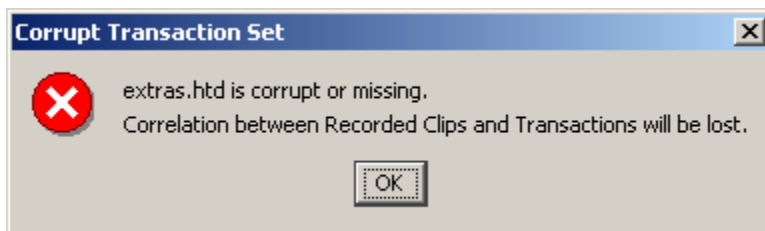
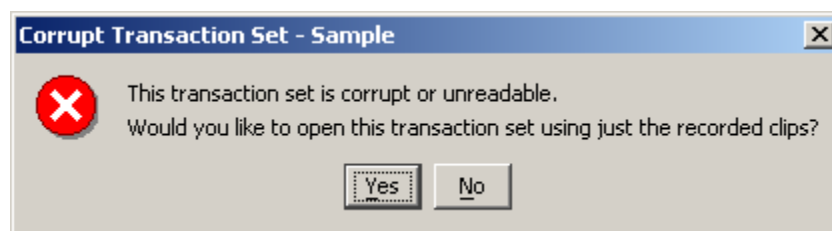
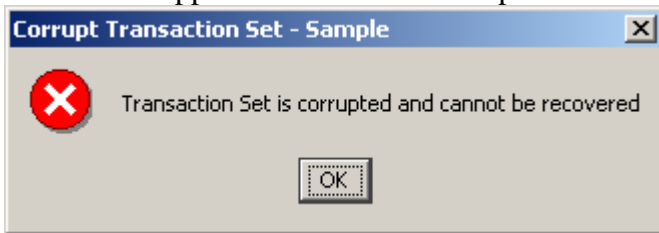


Figure 58: Corrupt Transaction Set

If for some reason the hosttrans.htd or screendefs.htd file become corrupted or deleted, then the Transaction Designer will attempt to recover your recorded clips for you. If your transaction set is corrupted, you will see the following message:



If the entire application becomes corrupted or unreadable, the following message will appear:



In this case the transaction set cannot be used under any circumstances.

The following screens are missing Aid Keys

When creating transactions, each screen must contain an AID Key in order to advance the transaction to the next screen (except in the case of a 5250 auto-enter). The last screen in a transaction and transactions containing a single screen never require AID Keys. When attempting to publish a transaction that has missing Aid Keys, you will receive the following warning.

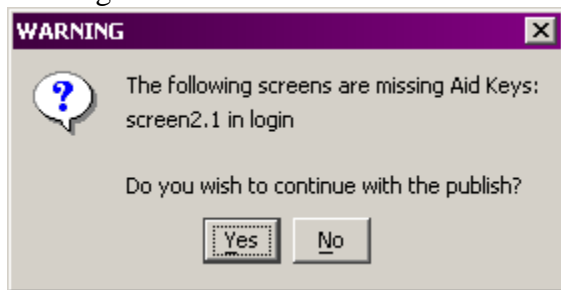
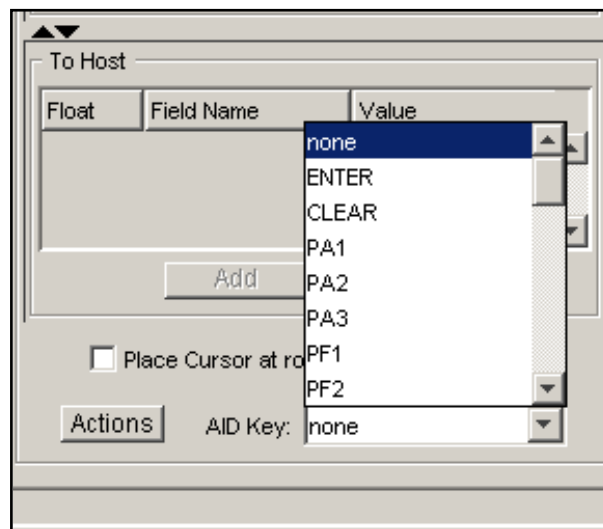


Figure 59: WARNING

The message informs the user which transaction(s) and screen(s) have a missing Aid Key violation. If necessary, navigate to the appropriate screen and simply choose an Aid Key from the drop down list.



Recorder Errors

Errors may occur when launching the Transaction Designer recorder. If the screen is blank, the connection to the host failed. Verify that the information for accessing the host is correct, and that the host sessions are active. Select **Session | Connect** to retry the connection.

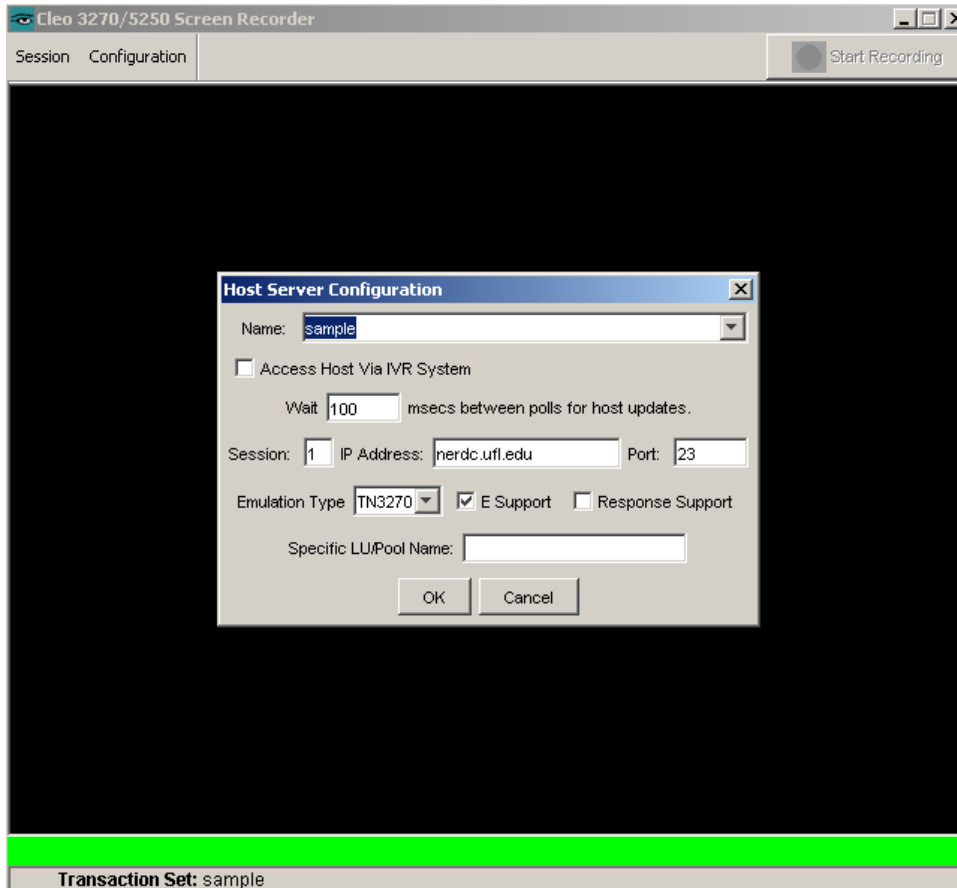


Figure 60: Host Server Settings Dialog

Session Id in Use

When running the Recorder and Validator at the same time, connecting to the same session id will cause the following error:

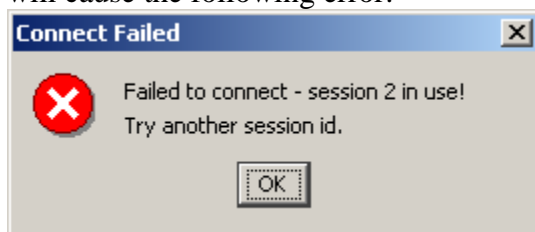


Figure 61: Session in Use

Retry the connection and specify a different session id in the range 1 – 10.

Restrictions

- Each screen in a transaction must have at least one identifier.
- Cannot “swipe” beyond the boundaries of an input field for a **To Host** field value.
- The sum of the lengths of all **From Host** fields defined in a screen cannot exceed 4000.

Appendix B: Sample Transaction Map

The following is a sample copy of the HTML file located on:

C: \Program Files\CleoTD\trans\sample

This file contains a list (i.e., “mapping”) of all host transactions and screen definitions currently defined for the sample transaction set.

For details on creating such a mapping, refer to the **File Menu** description, **Create Map** feature on page 45.

File contents:

Transaction Set: sample

Transaction: login

Screen: FirstScreen	
Base Screen:	FirstScreen
Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>NNN NNN EEEEEEEEEEE RRRRRRRRRR DDDDDDDDD</i>", startrow=1, startcol=10, endrow=1, endcol=59 Text present: "<i>Command or Service Name</i>", startrow=21, startcol=2, endrow=21, endcol=24 Text present: "<i>PF1=Help</i>", startrow=24, startcol=2, endrow=24, endcol=9
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> <i>field2</i>: startrow=21, startcol=31, endrow=21, endcol=79, float=no, value="luis"
Aid Key:	ENTER
Screen: LUISMenu	
Base Screen:	LUISMenu
Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>LUIS Menu</i>", startrow=1, startcol=2, endrow=1, endcol=10 Text present: "<i>Type the number of your choice</i>", startrow=22, startcol=2, endrow=22, endcol=31
From-Host Fields:	
To-Host Fields:	

Aid Key:	
-----------------	--

Transaction: park

No Screens Defined

Transaction: logout

Screen: AnyScreen1	
Base Screen:	null
Screen Identifiers:	<ul style="list-style-type: none"> Any Screen Accepted
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> <i>field21</i>: startrow=22, startcol=58, endrow=22, endcol=59, float=yes, value="30"
Aid Key:	ENTER
Screen: AnyScreen2	
Base Screen:	null
Screen Identifiers:	<ul style="list-style-type: none"> Any Screen Accepted
From-Host Fields:	
To-Host Fields:	
Aid Key:	

Transaction: recovery

No Screens Defined

Transaction: search

Screen: screen2.2	
Base Screen:	LUISMenu
Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>LUIS Menu</i>", startrow=1, startcol=2, endrow=1, endcol=10 Text present: "<i>Type the number of your choice</i>", startrow=22, startcol=2,

	endrow=22, endcol=31
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> • <i>field6</i>: startrow=22, startcol=58, endrow=22, endcol=59, float=no, value="8"
Aid Key:	ENTER

Screen: screen3.1

Base Screen:	screen3.1
Screen Identifiers:	<ul style="list-style-type: none"> • Text present: "<i>UNIVERSITY OF FLORIDA</i>", startrow=2, startcol=58, endrow=2, endcol=78 • Text present: "<i>NEXT COMMAND</i>", startrow=24, startcol=2, endrow=24, endcol=14
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> • <i>request</i>: startrow=24, startcol=16, endrow=24, endcol=78, float=no, value="a=steinbach"
Aid Key:	ENTER

Screen: Results

Base Screen:	Results
Screen Identifiers:	<ul style="list-style-type: none"> • Text present: "<i>Search Request</i>", startrow=1, startcol=2, endrow=1, endcol=15 • Text present: "<i>NEXT COMMAND</i>", startrow=24, startcol=2, endrow=24, endcol=13
From-Host Fields:	<ul style="list-style-type: none"> • <i>linesIto6</i>: startrow=4, startcol=2, endrow=8, endcol=79 • <i>NoEntries</i>: startrow=2, startcol=18, endrow=2, endcol=20 • <i>moreData</i>: startrow=19, startcol=51, endrow=19, endcol=59
To-Host Fields:	
Aid Key:	

Transaction: search_park**Screen: Results2**

Base Screen:	null
---------------------	------

Screen Identifiers:	<ul style="list-style-type: none"> Any Screen Accepted
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> <i>field11</i>: startrow=24, startcol=16, endrow=24, endcol=78, float=yes, value="sta"
Aid Key:	ENTER
Screen: LUISMenu2	
Base Screen:	LUISMenu3
Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>LUIS Menu</i>", startrow=1, startcol=2, endrow=1, endcol=10 Text present: "<i>Type the number of your choice</i>", startrow=22, startcol=2, endrow=22, endcol=31
From-Host Fields:	
To-Host Fields:	
Aid Key:	

Transaction: nextScreen

Screen: Results.1	
Base Screen:	Results
Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>Search Request</i>", startrow=1, startcol=2, endrow=1, endcol=15 Text present: "<i>NEXT COMMAND</i>", startrow=24, startcol=2, endrow=24, endcol=13
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> <i>field4.1</i>: startrow=24, startcol=16, endrow=24, endcol=78, float=no, value="f"
Aid Key:	ENTER
Screen: Results3	
Base Screen:	Results3
Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>Search Request</i>", startrow=1, startcol=2, endrow=1, endcol=15 Text present: "<i>NEXT COMMAND</i>", startrow=24, startcol=2, endrow=24,

	endcol=13
From-Host Fields:	<ul style="list-style-type: none"> • <i>line1</i>: startrow=4, startcol=2, endrow=4, endcol=78 • <i>line2</i>: startrow=5, startcol=2, endrow=5, endcol=78 • <i>line3</i>: startrow=6, startcol=2, endrow=6, endcol=78 • <i>line4</i>: startrow=7, startcol=2, endrow=7, endcol=78 • <i>line5</i>: startrow=8, startcol=2, endrow=8, endcol=78 • <i>moreData</i>: startrow=19, startcol=51, endrow=19, endcol=59
To-Host Fields:	
Aid Key:	

Transaction: nextScreen_park

Screen: screen7.2	
Base Screen:	null
Screen Identifiers:	<ul style="list-style-type: none"> • Any Screen Accepted
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> • <i>field20</i>: startrow=24, startcol=16, endrow=24, endcol=78, float=yes, value="sta"
Aid Key:	ENTER

Screen: LUISMenu3	
Base Screen:	LUISMenu3
Screen Identifiers:	<ul style="list-style-type: none"> • Text present: "<i>LUIS Menu</i>", startrow=1, startcol=2, endrow=1, endcol=10 • Text present: "<i>Type the number of your choice</i>", startrow=22, startcol=2, endrow=22, endcol=31
From-Host Fields:	
To-Host Fields:	
Aid Key:	

Transaction: getAllScreens

Screen: Results.1.1	
Base Screen:	Results

Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>Search Request</i>", startrow=1, startcol=2, endrow=1, endcol=15 Text present: "<i>NEXT COMMAND</i>", startrow=24, startcol=2, endrow=24, endcol=13
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> <i>field4.1.1</i>: startrow=24, startcol=16, endrow=24, endcol=78, float=no, value="f"
Aid Key:	ENTER

Screen: Results3.1

Base Screen:	Results3
Screen Identifiers:	<ul style="list-style-type: none"> Text present: "<i>Search Request</i>", startrow=1, startcol=2, endrow=1, endcol=15 Text present: "<i>NEXT COMMAND</i>", startrow=24, startcol=2, endrow=24, endcol=13
From-Host Fields:	<ul style="list-style-type: none"> <i>lines1to6</i>: startrow=4, startcol=2, endrow=9, endcol=80 <i>moreData</i>: startrow=19, startcol=51, endrow=19, endcol=59
To-Host Fields:	<ul style="list-style-type: none"> <i>_COMMAND1</i>: startrow=0, startcol=0, endrow=0, endcol=0, float=no, value="[@RULE AND moreData = CONTINUED getAllScreens -@@-]"
Aid Key:	

Transaction: getAllScreens_park

Screen: screen7.2.1

Base Screen:	null
Screen Identifiers:	<ul style="list-style-type: none"> Any Screen Accepted
From-Host Fields:	
To-Host Fields:	<ul style="list-style-type: none"> <i>field20.1</i>: startrow=24, startcol=16, endrow=24, endcol=78, float=yes, value="sta"
Aid Key:	ENTER

Screen: LUISMenu4

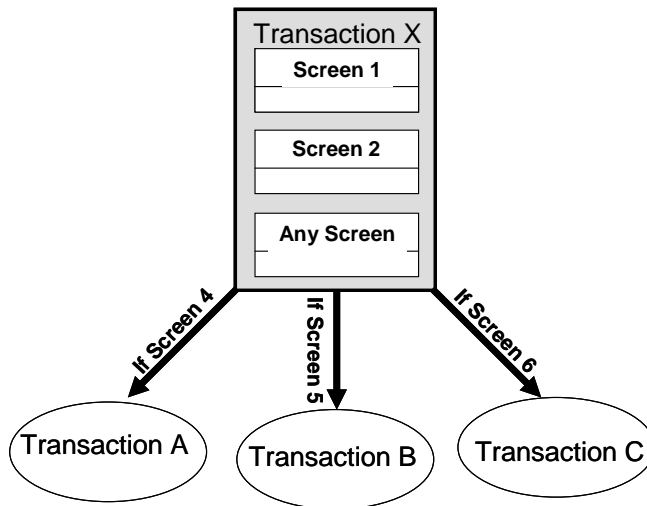
Base Screen:	LUISMenu3
---------------------	-----------

Screen Identifiers:	<ul style="list-style-type: none">• Text present: "<i>LUIS Menu</i>", startrow=1, startcol=2, endrow=1, endcol=10• Text present: "<i>Type the number of your choice</i>", startrow=22, startcol=2, endrow=22, endcol=31
From-Host Fields:	
To-Host Fields:	
Aid Key:	

Appendix C: Conditional Transactions

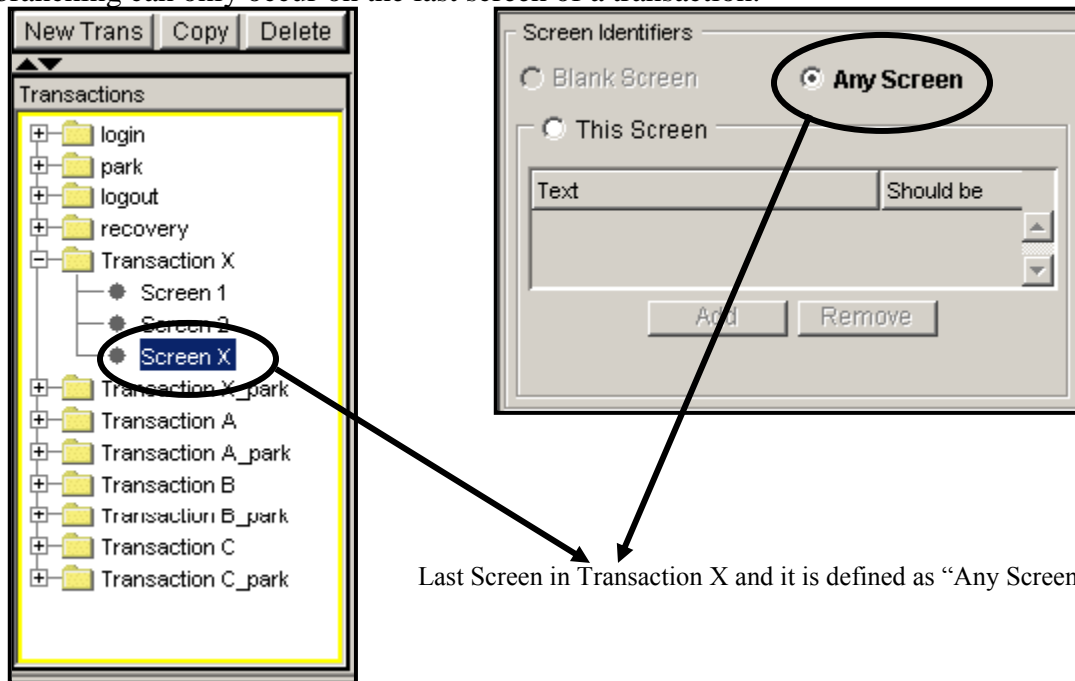
Creating Branches

In some cases, it may be useful to check what screen is present before determining which transaction to run next. For example, consider the following situation...



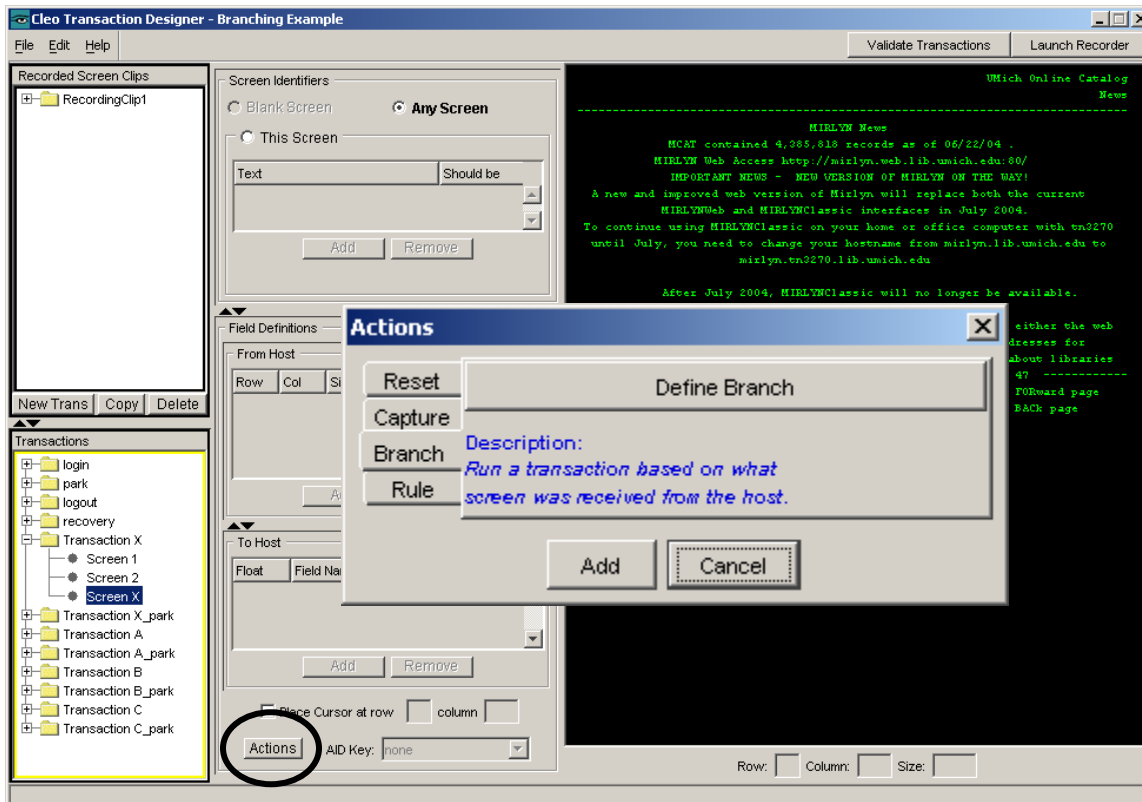
Transaction X is run which consists of Screen 1 followed by Screen 2, followed by an “**Any Screen**”. An “**Any Screen**” is exactly what it sounds like, a screen that contains anything, including a blank screen. In this situation, we need to detect what screen has arrived in order to determine which transaction to run next. In this case, if the “**Any Screen**” happens to be Screen 4, Transaction A is run. If it is Screen 5, Transaction B is to be run and Screen 6 should trigger Transaction C. This type of action is called “**Branching**” and can easily be done in the Transaction Designer. The following demonstrates how to create the above scenario step by step.

First, click on the screen in the transaction where the “**Branch**” is to occur. Keep in mind, that branching can only occur on the last screen of a transaction.

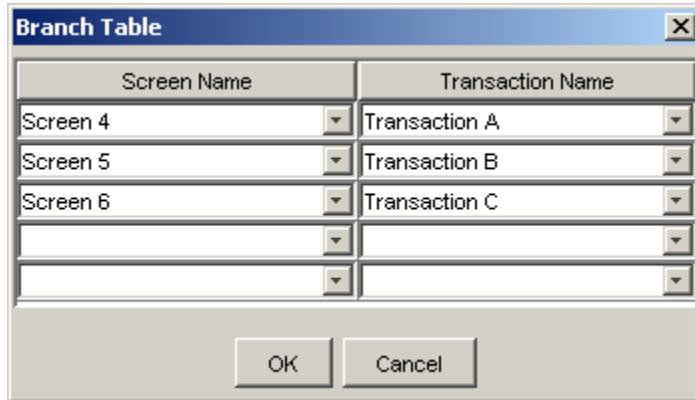


Last Screen in Transaction X and it is defined as “Any Screen”

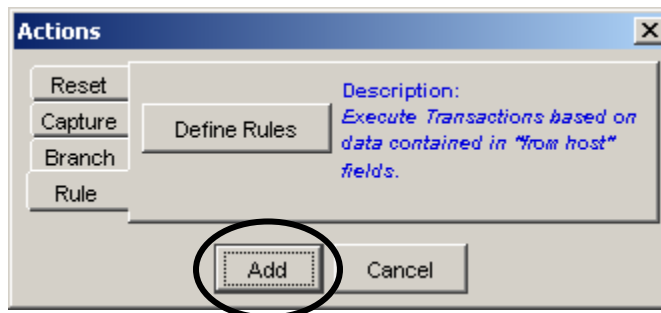
Click on the “**Actions**” button located near the bottom left of the Transaction Designer. Upon clicking the “**Actions**” button the “**Actions**” Dialog box will appear. Click on the “**Branch**” tab and then click “**Define Branch**” .



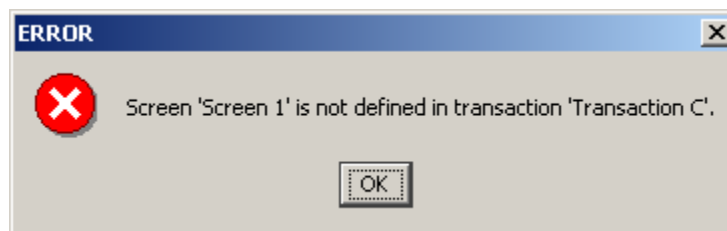
The “**Branch Table**” dialog will appear. (Note: the “Actions” Dialog box will remain while you create the branch in the “**Branch Table**”).



The column on the left contains the expected screen and the column on the right contains the transaction to be run. In this case, if Screen 4 is encountered as the last screen in this transaction then Transaction A will be run. When finished creating the Branch, click on “**OK**” and then click “**Add**” on the “Actions” dialog box.



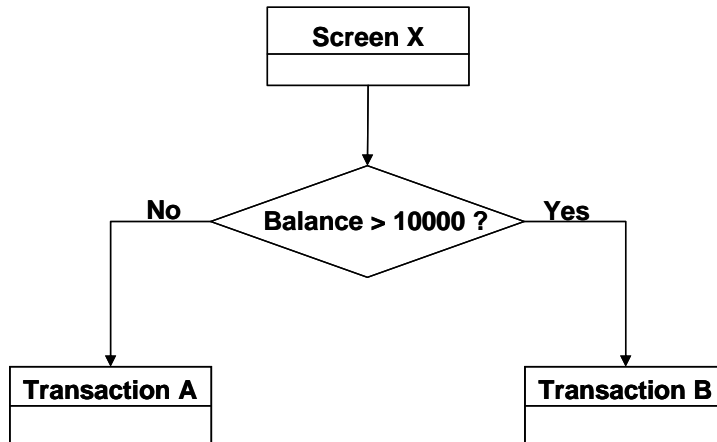
If a Screen Name is selected on the left that does not appear in the Transaction selected on the right you will see the following message:



This means that ‘Screen 1’ is not contained or does not exist within ‘Transaction C’, so this scenario is invalid.

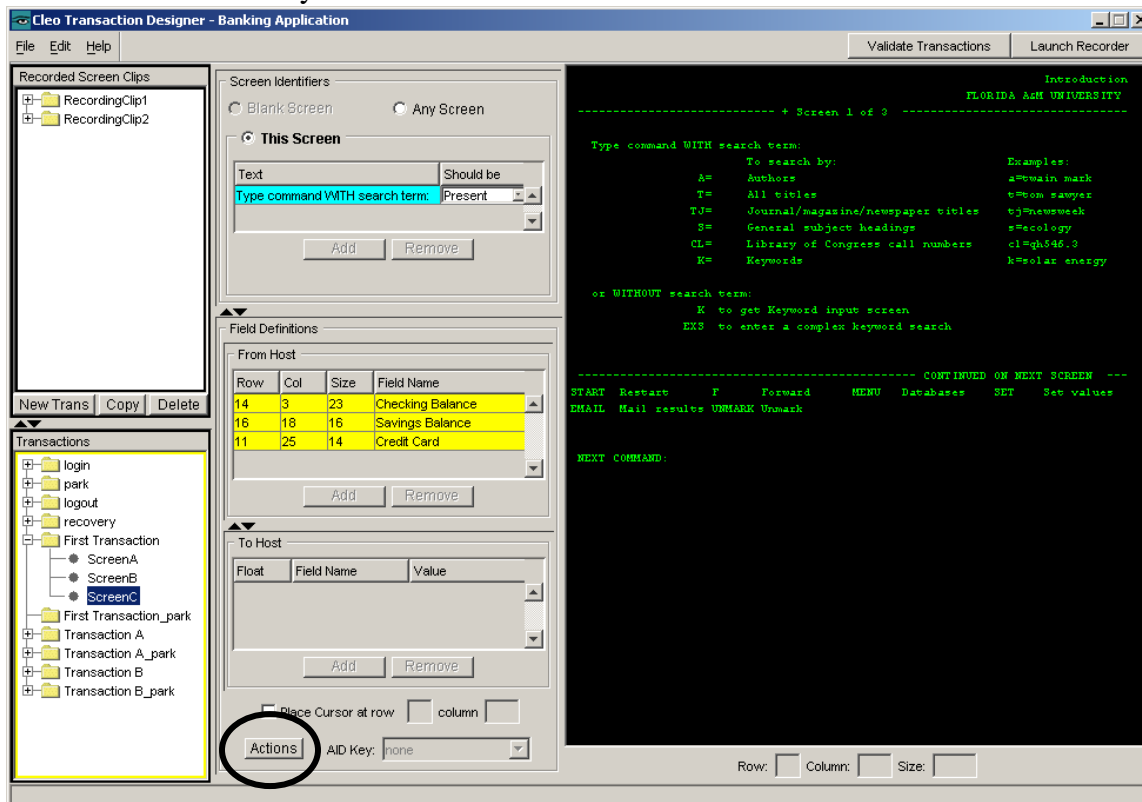
Creating Rules

In some cases, it may be necessary to check the value of a field from the host before determining which transaction to run next. For example, consider the following situation.

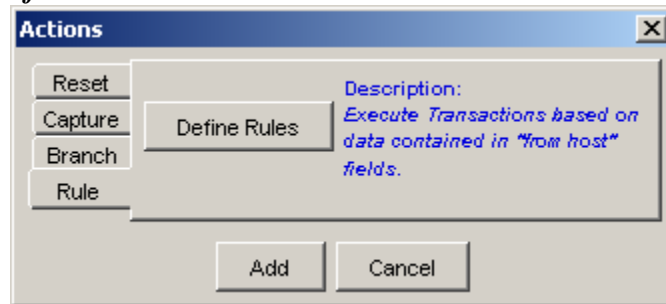


The user arrives at the Screen X. If the account balance is > 10000 Transaction B is run, otherwise transaction A is run. This type of scenario can be easily accomplished in the Transaction Designer by using the “*Rule Builder*” to define the conditions. The following is a step by step guide to creating rules using the “*Rule Builder*”.

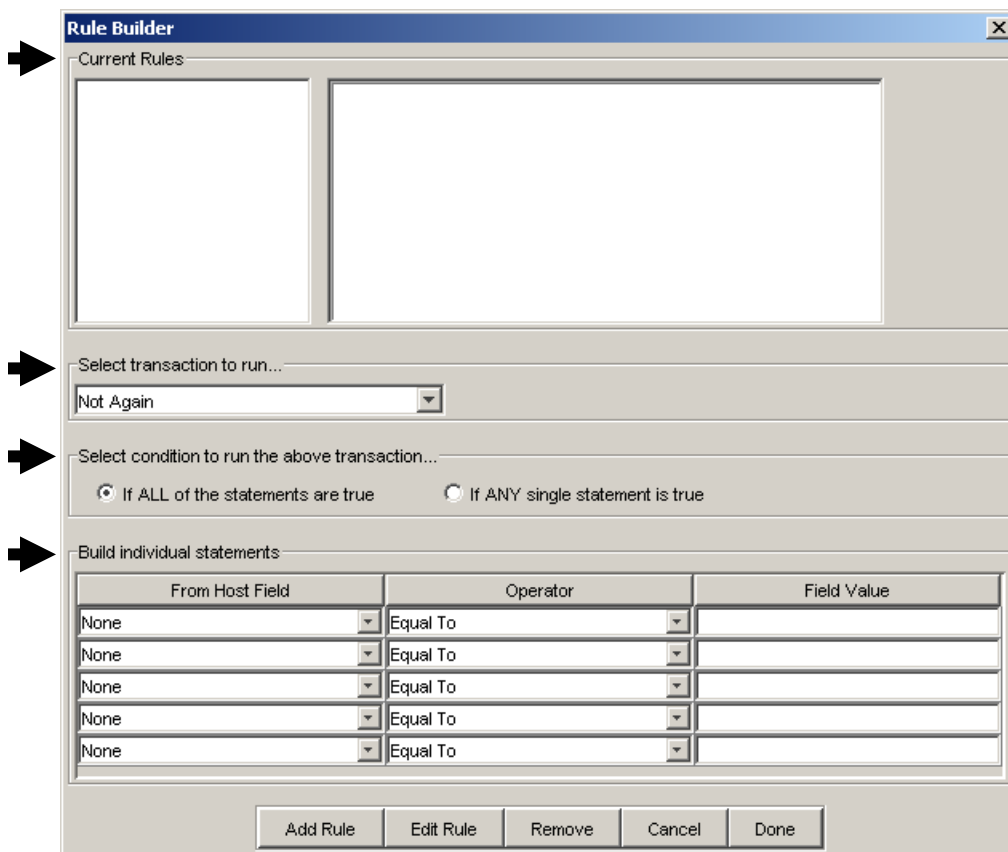
Click on the “*Actions*” button located near the bottom left of the Transaction Designer. Keep in mind that rules can only be created on the LAST screen of a transaction.



Upon clicking the “**Actions**” button the “**Actions**” Dialog box will appear. Click on the “**Rule**” tab and then click “**Define Rules**”



The “**Rule Builder**” dialog will appear. (Note: the “**Actions**” Dialog box will remain while you create rules in the “**Rule Builder**”).



The “**Rule Builder**” is divided into 4 sections. The top section contains 2 window panes and is titled “**Current Rules**”. This is simply a display window that shows rules as they are defined. The next section contains the transaction to be run if the conditions are met and is titled “**Select transaction to run...**” The section titled “**Select condition to run the above transaction**”

determines if the transaction will be run based on ALL of the statements being true, or ANY of the statements being true. Finally, the bottom section titled “**Build individual statements**” is where the individual statements are created.

Rule Builder

Current Rules

Select transaction to run...
Transaction A

Select condition to run the above transaction...
 If ALL of the statements are true If ANY single statement is true

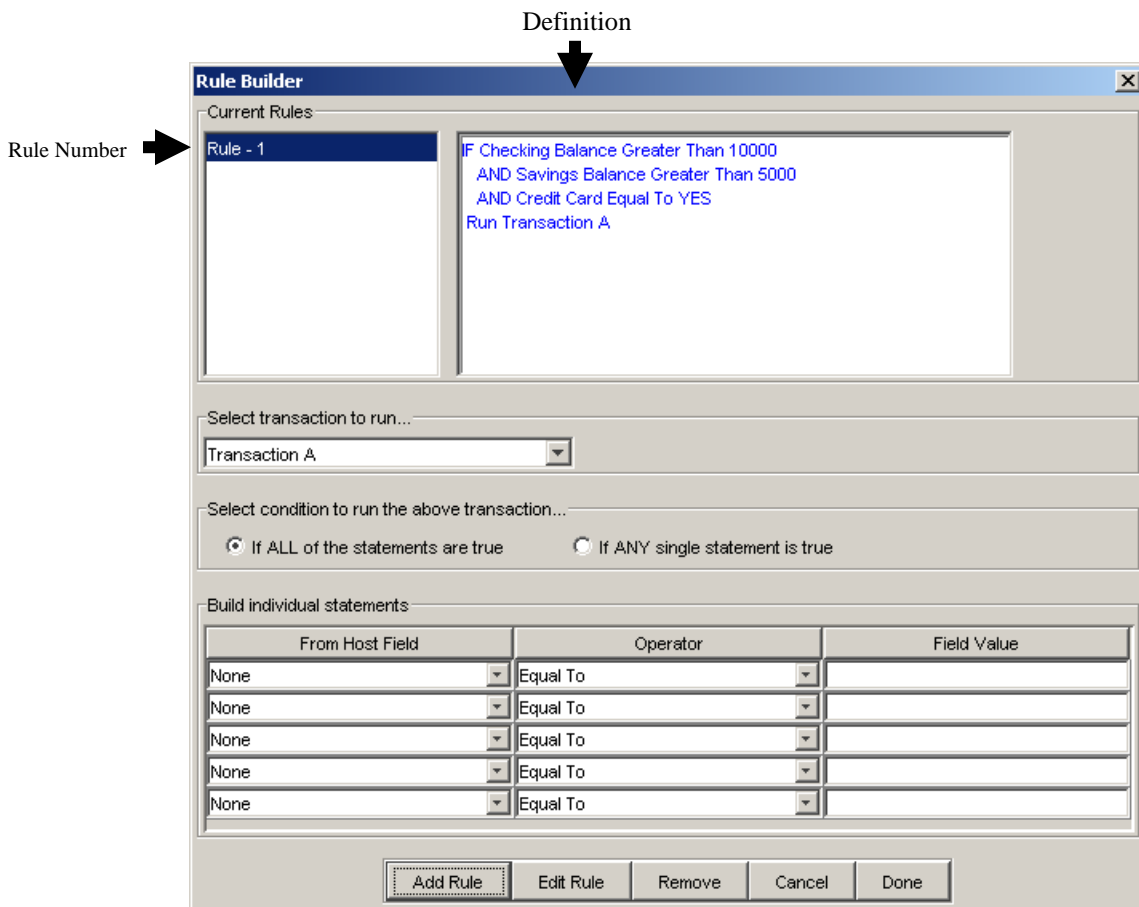
Build individual statements

From Host Field	Operator	Field Value
Checking Balance	Greater Than	10000
Savings Balance	Greater Than	5000
Credit Card	Equal To	YES
None	Equal To	
None	Equal To	

Add Rule Edit Rule Remove Cancel Done

Since conditional transactions are based upon values contained in the “**From Host**” fields on the last screen of a transaction, select the “**From Host Field**” in the far left column that corresponds to the field you wish to check. In the first row, notice that “**Checking Balance**” is selected. The middle column is the “**Operator**” and in this case “**Greater Than**” is selected. The Right column is the value we will compare to what is received from the host. Up to 5 “**Statements**” can be added to each Rule and there is no limit to the number of rules you can create.

Click on “**Add Rule**” when you are finished.



A Rule named “**Rule – 1**” has been added to the “**Current Rules**” window and its definition is on the right. Notice the appearance of the word “AND”. Since the radio button “**if ALL Of The Following Statements Are True**” was selected, every statement must be true in order for “**Transaction A**” to be run.

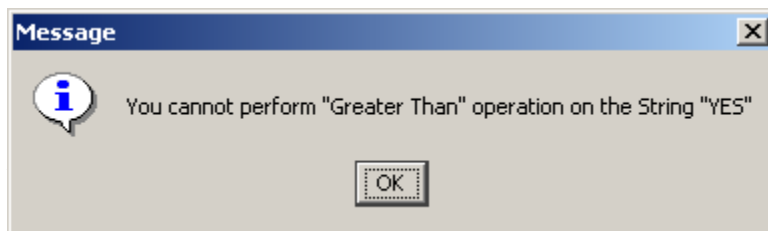
Checking Balance must be Greater Than 10000

AND Savings Balance must be Greater Than 5000

AND Credit Card must be Equal To YES

If all of these conditions are true, then Transaction A will be run.

Keep in mind, when building individual statements you can only perform “Equal To” or “Not Equal To” operations on a string. For example, you could not create a statement that reads **Credit Card Greater Than YES**. If you attempt to do so, you will see the following error.



Rule Builder

Current Rules

Rule - 1
Rule - 2

IF Checking Balance Less Than/Equal To 10000
OR Savings Balance Less Than/Equal To 50000
OR Credit Card Equal To NO
Run Transaction B

Select transaction to run...
Transaction B

Select condition to run the above transaction...
 If ALL of the statements are true
 If ANY single statement is true

Build individual statements

From Host Field	Operator	Field Value
None	Equal To	
None	Equal To	
None	Equal To	
None	Equal To	
None	Equal To	

Add Rule Edit Rule Remove Cancel Done

This time, the “*if ANY Of The Following Statements Are True*” radio button was checked. This means

IF Checking Balance is Less Than or Equal to 10000

OR Savings Balance is Less Than or Equal to 5000

OR Credit Card is Equal to NO

If any of the above are true, Then “*Transaction B*” will be run.

You may add as many rules to the window as you wish before clicking the “*Done*” button. Notice that each time you add a rule it is given a number. When the *Transaction Processor* processes these statements, it will run the transaction for the first rule it encounters that is true. For example, if “*Rule – 1*” is true in this case, then Transaction A will be run regardless of whether or not “*Rule – 2*” also happens to be true.

You may also edit a set of Rules by clicking on the “*Edit Rule*” button.

The text in the “Rules Created” window will turn red and the rule will be repopulated in all of the appropriate places. The areas in orange highlight the various components of the rule created. Simply make the appropriate changes and click on the “*Save*” button. The “*Rule Builder*” window will return to its normal state.

When finished, click the “*Done*” Button and then Click on the “*Add*” button in the “*Actions*” dialog box. The entire set of Rules is processed into a one long string and is placed in the “*To Host*” field table in the *Transaction Designer*. You may reopen this set of rules any time by simply double clicking on it.

In summary...

The "**Rule Builder**" is a tool where a set of "**Rules**" is created.

A set of "**Rules**" can have an unlimited number of "**Rules**" (e.g. Rule – 1, Rule – 2)

A "**Rule**" can contain up to 5 "**Statements**" (e.g. Checking Balance Greater Than 10000)

Appendix D: Publishing VoiceXML

An option of the Transaction Designer is the Publish VoiceXML feature which is accessed through the VoiceXML Generator tool. VoiceXML code generated by the VoiceXML Generator can be run. However, the VoiceXML Generator is not intended to be a source code editor. Therefore, additional processing steps (such as prompts and validation) must be added using a separate VoiceXML development tool.

VoiceXML Generator

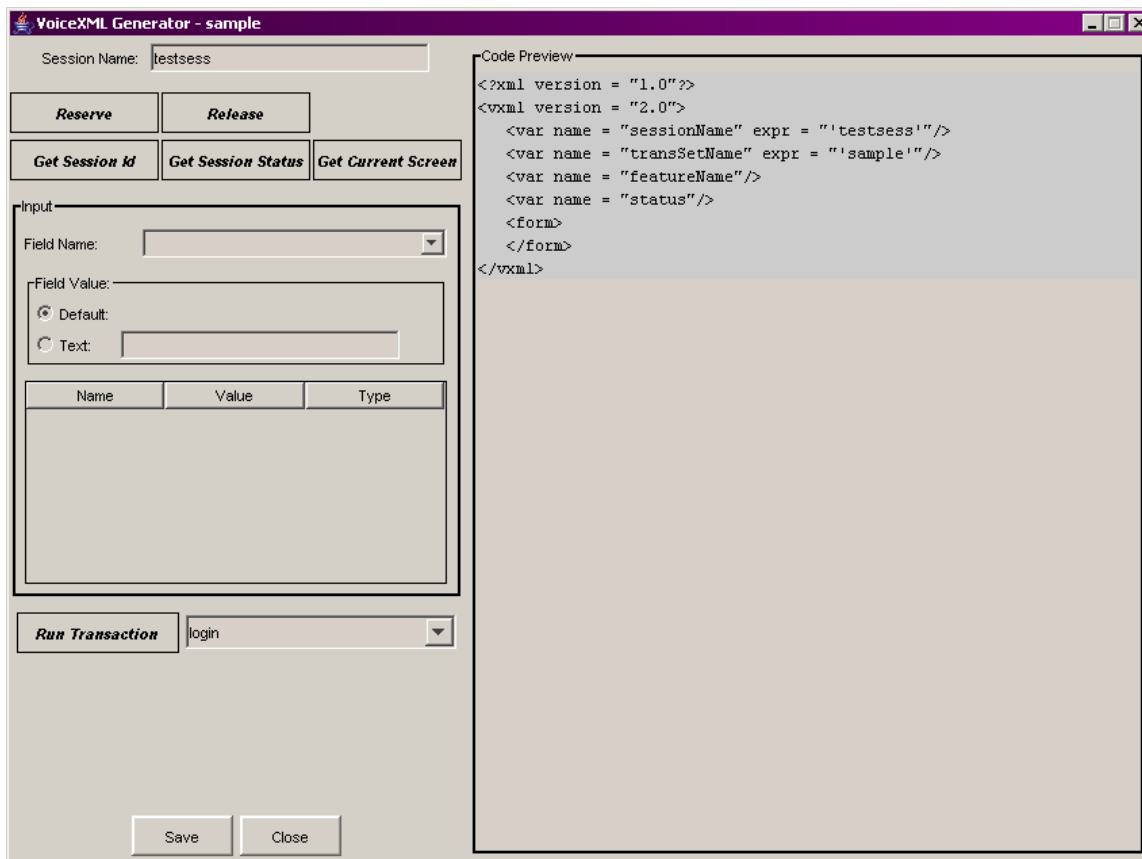


Figure 64 - VoiceXML Generator

The VoiceXML Generator can be launched from the Transaction Designer **File | Publish | VoiceXML** menu.

The most recently published transaction set will be loaded into the VoiceXML Generator and its name will be displayed in the top application bar.

The associated transactions will be listed in the drop-down box next to the *Run Transaction* button.

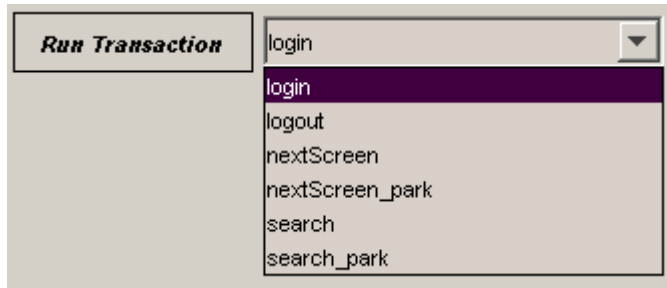


Figure 65 – Transactions

Functions Provided

Within any VoiceXML application, the basic Transaction Processor functions (reserve, release, getSessionId, getSessionStatus, getCurrentScreen, and runTransaction) can be invoked using the VoiceXML Connector. The VoiceXML Connector is a separately licensed and feature of the Transaction Processor that provides VoiceXML applications access to the Transaction Processor through the use of subdialogs. (See the Transaction Processor Programmer's Guide for more details.) The VoiceXML Generator aids in the proper formation of the subdialog for each function.

Subdialogs can be added to the VoiceXML application by dragging and dropping the required function from the left (function) panel to the Code Preview panel.

To drag and drop a function from the function panel to the Code Preview panel:

1. Click on the *function* button (do not release the mouse button).
2. Drag the cursor from the *function* button to the Code Preview panel (while continuing to hold down the mouse button).
3. Release the mouse button when a black line is displayed in the Code Preview panel at the appropriate insertion location.

```
Code Preview
<?xml version = "1.0"?>
<vxml version = "2.0">
  <var name = "sessionName" expr = "'testsess'"/>
  <var name = "transSetName" expr = "'sample'"/>
  <var name = "featureName"/>
  <var name = "status"/>
  <form>
  </form>
</vxml>
```

4. The Code Preview will then display the inserted code and the associated variables.

```
Code Preview
<?xml version = "1.0"?>
<vxml version = "2.0">
  <var name = "sessionName" expr = "'testsess'"/>
  <var name = "transSetName" expr = "'sample'"/>
  <var name = "featureName"/>
  <var name = "status"/>
  <form>
    <var name = "returnCode"/>
    <block>
      <assign name = "featureName" expr = "'reserve'"/>
    </block>
    <subdialog name = "reserve_0" src = "CleoHostAccess" namelist =
      <filled>
        <assign name = "status" expr = "reserve_0.status"/>
        <assign name = "returnCode" expr = "reserve_0.returnCode"/>
      </filled>
    </subdialog>
    <block>
      status is <value expr = "status"/>
      return code is <value expr = "returnCode"/>
    </block>
  </form>
</vxml>
```

Reserve, Release, Get Session Id, Get Session Status, Get Current Screen

Reserve, Release, Get Session Id, Get Session Status, and Get Current Screen are added to the Code Preview panel without the need to manually configure any parameters. The parameters are set programmatically and cannot be overwritten.

Run Transaction

Input

Field Name:

Field Value:

Default: mcat

Text:

Name	Value	Type
DBSelect	mcat	Default
request	a=steinbach	Default

Run Transaction

Figure 66 - Run Transaction Configuration

The only function that may require configuration of parameters is *Run Transaction*. The parameters that are set before Run Transaction can be added to the Code Preview panel are:

- **Transaction** – transaction to be run
- **Input** – To-Host/input field(s) (as defined in the Transaction Designer) associated with the transaction that are assigned values(s):
 - Default – value originally assigned to the field in the Transaction Designer
 - Text – overrides the default value with the entered value

To set the transaction:

Select the required transaction from the drop-down list next to the *Run Transaction* button.

To set input fields:

1. Select the field from the Field Name drop-down list.
2. The Field Value panel will display the currently set value of the selected field.
3. Select either Default (to accept the default value) or Text (to enter a different value). If Text is selected, enter a value into the Text field. The value that is selected or entered will appear in the Field Name/Value/Type table.

Glossary

AID key: 3270 Attention Identification key, when pressed, causes data to be sent to the host. Examples include ENTER, PA1, PA2, PA3, PF1-PF24, CLEAR.

Base Screen: Transaction screen containing text identifiers which can be used by other transaction screens for the purpose of screen identification.

Field definition: specification of an area on the host screen image, including its name, start row and column, length, and whether it is input or output.

Host Screen: A 3270 or 5250 host application screen

Host transaction sequence: A representation of a series of host application screens. The following are pre-defined transaction names:

Login: used to log on to the host

Park: used to navigate to a specific host application screen

Logout: used to log off the host

Recovery: used to navigate from an unknown or erroneous application screen

Input field: area on a 3270/5250 screen containing data sent to the host.

IR: Interactive Response system, which automatically responds to telephone requests for information.

IR Application: Any application developed using IVR Designer or another IR development tool.

Keyboard macro: variable text stored in a table to be used as input in a host transaction sequence. Each entry is referenced by column number when defining “to host” field definitions.

Output field: area on a 3270/5250 screen containing data received from the host, to be sent to the IVR application.

Partial Publish: When using the publish function to create the XML files for screen definitions and transactions, the XML file could not be generated for any transaction containing screens with no identifiers.

Recording clip: information describing a contiguous series of host screen images with associated keyboard input used to progress from one screen to the next. The last screen image in the clip does not have an associated AID key.

Re-Park transaction: a user-defined transaction name with a suffix of “_park”, such as “transaction1_park”. The re-park transaction is run by the plug-in after its companion user-

defined transaction has been successfully run. See the “*Cleo 3270 Plug-in Programmer's Guide*” or “*Transaction Processor Programmer's Guide*” for a more detailed discussion.

Right click: the action of using the alternate mouse button.

Screen definition: characteristics identifying a host application screen including locations on the screen to be used in identifying the screen (screen identifiers), input fields, and output fields.

Screen identifier: an area on the host screen image to be used in identifying the screen, specified by its start row and column, and text.

Transaction Set: a collection of transactions and screen definitions used by the Transaction Processor or Cleo Plug-in. A transaction set may be assigned to one or more sessions. For example if “tset1” is assigned to sessions 2-11, “tset2” is assigned to sessions 12-31, the IR application “ir1” which runs “tset1” is assigned to channels 0-9, the IR application “ir2” which runs “tset2” is assigned to channels 10-19, and the IR application “ir3” which also runs “tset2” is assigned to channels 20-29, then when a call is made into channel 1, “tset1” will be run. Likewise, if a call is made into channel 11 or channel 25, “tset2” will be run. (See also Application)

User-defined transaction: contains host transaction sequence that is run by the Transaction Processor or plug-in when requested by the IR application to retrieve information from the host. An empty user-defined transaction named “transaction1” is created by the Transaction Designer.

VoiceXML: markup language for creating voice user interfaces. It uses speech recognition and/or touchtone (DTMF keypad) for input, and pre-recorded audio and text-to-speech synthesis (TTS) for output. Callers interact with VoiceXML applications via a VoiceXML “interpreter” (also known as a “browser”) running on a telephony server.

XML: short for *Extensible Markup Language*, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents.

Index

3270	5	macros	27, 43
actions	28	No login transaction	51
Administration GUI	5	Open	45
AID key	6	Overlapping Fields	51
Component Diagram	6	port	16
configurator functions	16	Publish	45
Create Map	45	Recorded Screen Clips	18
default settings	16	Recorder Errors	55
Define Fields	25	recording screen images	16
definitions	78	recording, start and stop	17
Delete Recording Clip	46	redraw	15
Duplicate Fields	51	Rename	46
Duplicate screen definitions	34	restrictions	56
edit menu	46	sample transaction map	57
emulation type	12	Save	45
Exit	45	screen definitions	22
fields received from Host	25	session	12
Fields Sent to the Host	25, 26	start and stop recording	17
File Menu	45	Sum of field lengths >4000	50
fromHost field lengths >4000	50	Swiped area beyond field	49
General Errors	47	table driven input fields	27
Help Menu	46	Transaction Designer	5, 8
Highlight Input Field Boundaries	26, 46	Transaction Processor	5
HLLAPI	5	Transfer XML	45
Host Screen Recorder	12	troubleshooting	47
key features	6	Validate Transactions	See Validator.
keyboard macro table	27	Validator	7, 37
keyboard map	16	wait	12, 16
List Related Screens	46		