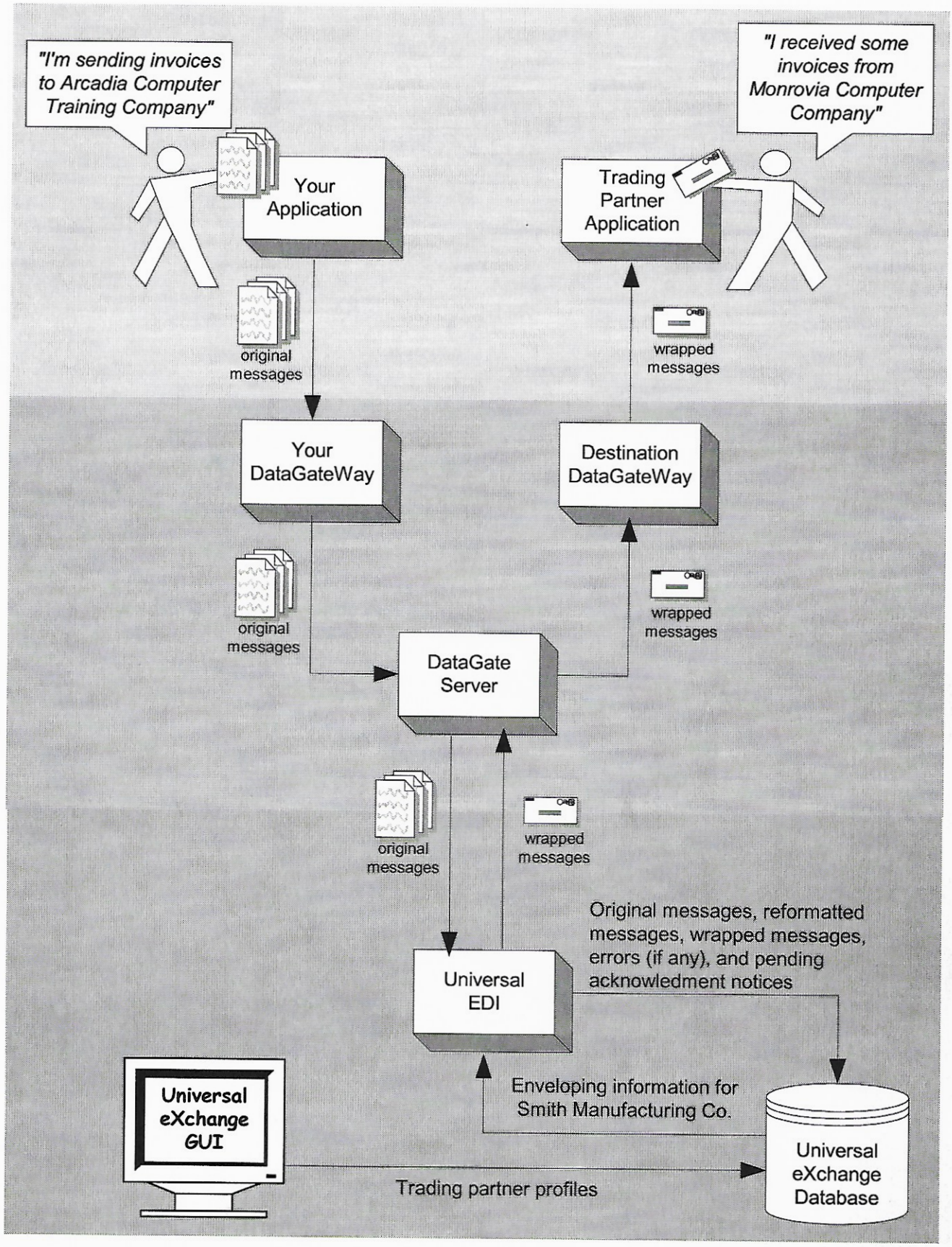
"I'm sending invoices to Arcadia Computer Training Company"
"I received some invoices from Monrovia Computer Company"
Your Application
Trading Partner Application
original messages
wrapped messages
Your DataGateWay
Destination DataGateWay
original messages
wrapped messages
DataGate Server
original messages
wrapped messages
Original messages, reformatted messages, wrapped messages, errors (if any), and pending acknowledment notices
Universal EDI
Universal eXchange GUI
Enveloping information for Smith Manufacturing Co.
Universal eXchange Database
Trading partner profiles

What are the Universal eXchange DataGateWays?

Universal eXchange consists of seven DataGateWays, each of which performs a specific EDI message handling function.

- **Transaction Processing DataGateWays**
  The Transaction Processing DataGateWays handle messages using information previously stored about a particular trading partner in the UX database. Four instances of the Transaction Processing DataGateWays are used to process:
  - Groups of inbound messages (batch transmission mode)
  - Groups of outbound messages (batch transmission mode)
  - Individual inbound messages (interactive transmission mode)
  - Individual outbound messages (interactive transmission mode)

- **Acknowledgment Monitor DataGateWay**
  The Acknowledgment Monitor DataGateWay monitors message acknowledgments that are overdue form trading partners and tries to resend the original transactions, as needed, to the appropriate trading partners. If acknowledgments continue to be overdue, this DataGateWay subsequently logs information about the transaction and corresponding error information in the Universal eXchange database. You can use the Universal eXchange GUI to view information about these logged transactions.

- **Transaction Polling DataGateWay**
  The Transaction Polling DataGateWay monitors the Universal eXchange transaction queue for outbound messages pending interactive transmission to your trading partners and routes them to your DataGate Server.

- **Batch Message Packager DataGateWay**
  The Batch Message Packager DataGateWay prepares outbound messages pending batch transmission to your trading partners for routing to your DataGate Server. This DataGateWay creates the header and trailer segments needed for each trading partner and wraps that information around each message.

Figure 2-2 illustrates how these DataGateWays work together to form the Universal eXchange solution.

Process Acknowledgement *es-ack-handler*

Purpose

The **es-ack-handler** API processes an acknowledgment sent by a trading partner that is in response to an outbound, original message from your application.

Description

The **es-ack-handler** API does the following:

- Receives a database connection handle and a list of information about the acknowledgment from the calling process.
- Updates the es_mtrk_outb and es_waiting_ack tables based on the transaction information retrieved by the es-init-trans API for the original message type. Specifically, es_mtrk_outb.ack_tm is updated and the associated row is deleted from es_waiting_ack.
- If provided, then the acknowledgment is also stored in the es_msg_storage table with a pointer to that table placed in es_mtrk_outb.ack_msg_id.
- If the acknowledgment tran_set_id does not match the expected tran_set_id and the ack_type = 'P', then the acknowledgment is ignored.
- Returns a value to the calling process that indicates whether

Usage

Use the **es-ack-handler** API for inbound transactions.

Syntax and Parameters

```
(es-ack-handler <connection_handle> <ack_stat> <ack>)
```

Parameter	Description
connection_handle	The previously established connection to the database. Required parameter Type: connection_handle
ack_stat	Information about the acknowledgment. Required parameter Type: list *(ack_tm ack_type tran_set_id level i_control_num g_control_num ts_control_num unique_id error_data)*

Parameter	Description
	List member — **Description**
	ack_tm — The date and time(***yyyymmddhhmmss*** format) Optional list member
	ack_type — Identifies the kind of acknowledgment (positive or negative).

This value ...	Specifies this kind of acknowledgment ...
P	Positive acknowledgment
N	Negative acknowledgment

	tran_set_id — The transaction set identification number for the acknowledgment.
	level — Specifies the level of the original message.

This value ...	Specifies this message level ...
I	Interchange
G	Group
T	Transaction

i_control_num	The interchange control number for the original message. Optional list member
g_control_num	The functional group control number for the original message. Optional list member
ts_control_num	The transaction set control number for the original message. Optional list member
unique_id	A unique identification for the original message. Optional list member
error_data	Error information. Optional list member

Return Values

The **es-ack-handler** API returns one of the following values.

This value is returned ...	If this occurs ...
#t (true)	The acknowledgement processed successfully.
#f (false)	The acknowledgement did *not* process successfully. Use the **es-get-error-str** API to retrieve the corresponding error message.

Example

The following Monk script example calls the es-ack-handler API with the assumption that the es-init-trans API executed successfully for the given acknowledgment. Also, a connection to the database, conn-handle, has been established before the API es-ack-handler is called. The es-ack-handler API uses the unique-id "270705" to find the appropriate row to update in the es_mtrk_outb table.

If successful, then the ack-msg value is stored in the es_msg_storage table and the msg_parent_id value is placed in the outbound message acknowledgement ID in the same row as the original message. The Ack-code value is set to "Positive" and the ack-tm value is set to the current time; both of these values are stored in the es_mtrk_outb table.

If the es-ack-handler API fails, then the error retrieved by the es-get-error-str API is displayed. Also, if the tran-set-id value does not match the expected value, then this acknowledgment will be ignored because it is positive.

```
(define ack-stat (list ""  ;ack-tm
                       "Positive"  ;ack-code
                       "P" ; ack-type
                       "997" ; tran-set-id
                       "t"  ;level
                        "" ;i-control-num
                        "" ;g-control-num
                        "" ;ts-control-num
                        "270705" ;unique-id
                        "" ; error-data
                    );list
)

(define ack-msg "ST*997*1145~AK1*HS*5678~AK2
*270*000000705~AK5*A~AK9*A*1*1*1~SE*6*1145")

(if (not (es-ack-handler conn-handle ack-stat
           ack-msg))
   (begin
      (display  "es-ack-handler failed!\n")
      (display (es-get-error-str))
      (newline)
   )

      (display "es-ack-handler succeeded!\n")
) ; if
```